Full Papers
**{ENTER}ing the Time Series {SPACE}: Uncovering the Writing Process through Keystroke Analyses**
Laura K. Allen¹, Matthew E. Jacovina¹, Mihai Dascalu², Rod D. Roscoe¹, Kevin M. Kent¹, Aaron D. Likens¹, & Danielle S. McNamara¹
¹Arizona State University, Tempe, USA  
²University Politehnica of Bucharest, Bucharest, Romania
{LauraKAllen, Matthew.Jacovina}@asu.edu, {Mihai.Dascalu}@cs.pub.ro, {Rod.Roscoe, Kdent4, Aaron.Likens, dsmcnama}@asu.edu

**ABSTRACT**
This study investigates how and whether information about students’ writing can be recovered from basic behavioral data extracted during their sessions in an intelligent tutoring system for writing. We calculate basic and time-sensitive keystroke indices based on log files of keys pressed during students’ writing sessions. A corpus of prompt-based essays was collected from 126 undergraduates along with keystrokes logged during the session. Holistic scores and linguistic properties of these essays were then automatically calculated using natural language processing tools. Results indicated that keystroke indices accounted for 76% of the variance in essay quality and up to 38% of the variance in the linguistic characteristics. Overall, these results suggest that keystroke analyses can help to recover crucial information about writing, which may ultimately help to improve student models in computer-based learning environments.

**Keywords**
Intelligent Tutoring Systems; Writing; Natural Language Processing; Feedback; Keystrokes; Temporality

1. INTRODUCTION
Effective written communication is a complex socio-cognitive skill that is important for success in academic and professional settings [1-2]. The writing process relies on both lower- and higher-level knowledge and skills, ranging from knowledge of the language and domain to strategies necessary for generating inferences and flexibly adapting to different task demands [1; 3-5]. Not surprisingly, then, the development of strong writing skills is extremely difficult and students consistently underachieve on national and international assessments of writing [6-8].

The remediation of these writing deficits is a similarly challenging task. The development of writing proficiency demands that students have access to high-quality instruction that is attuned to their particular needs. Research on writing instruction finds that students attain the greatest benefits when they are provided strategy instruction, practice, and feedback [9-10]. In particular, **deliberate practice** is crucial for the development of writing skills [11] and has been shown to help students regulate the planning, drafting, and reviewing stages of writing [10]. This type of meaningful and mindful practice inherently relies upon individualized formative feedback—feedback that reveals and explains actionable steps that students must take to improve. However, in large classrooms, detailed and targeted feedback on multiple essay drafts per student presents a daunting challenge for teachers.

Computer-based tools such as automated writing evaluation (AWE) systems have been developed to alleviate some of the pressures facing writing instructors [12]. At their core, AWE tools implement natural language processing (NLP) and machine learning techniques to accurately model the scores that expert human raters would assign based on the structure and content of students’ essays [13-14]. Additionally, many AWE systems and intelligent tutoring systems (ITSs) incorporate instructional elements such as lessons and practice games [15-16]. These modern systems extend beyond the assessment of essay quality to provide students with personalized feedback and recommendations for improvement.

Although a wealth of research has been conducted to validate the **accuracy** of AWE scores, much less attention has been paid to the pedagogical and rhetorical elements of these systems. Specifically, critics often cite the lack of sensitivity to different audiences, rhetorical moves, and writing processes as serious areas of concern, which can lead to impersonal and ineffective instruction and feedback [17;18]. These critiques are valid and point to much needed future research. Accordingly, researchers and developers have begun to re-focus their efforts away from establishing the accuracy of scoring models and towards the improvement of the personalized and nuanced aspects of the feedback and instruction.

To better detect and respond to differences among students’ writing processes and behaviors, we may need to embed assessments that are based on more than their written products and essay scores. These measures can be either visible or hidden from users (i.e., “stealth assessments”) [19], and can inform specific instruction and feedback that is tailored to students’ individual habits. In the context of computer-based learning environments, these assessments can be informed by a wealth of information that is easily logged within the system. Snow and colleagues (2014) [20], for example, developed stealth assessments of self-regulation within a reading comprehension tutoring system. They found that the predictability of students’ choices in the system was...
indicative of their self-regulation skill and influenced their performance on the learning task. Overall, such assessments may offer a viable solution to the writing process assessment problem. Both simple measures (e.g., typing speed) and complex measures (e.g., trajectories of mouse movements) might allow us to model the writing processes and characteristics of student users.

In this paper, we examine the efficacy of behavioral measures that are accessible (but rarely collected or analyzed) in writing training systems to detect information about students’ performance on their essays. In particular, we examine whether basic and time-sensitive keystroke indices can be used to model the scores and linguistic features of students’ essays. Our ultimate goal is to use these models to provide more individualized tutoring and feedback to students.

1.1 Keystroke Analyses for Writing

Keystroke data presents a potentially valuable approach for modeling students’ writing behaviors [e.g., 21]. Although researchers have made significant strides in leveraging the linguistic features of texts to understand writing quality, there has been substantially less research on students’ online or real-time writing processes. Due to challenges of data collection, prior writing research has focused primarily on students’ finished writing products and not their moment-by-moment writing processes. Recently, however, keystroke logging tools (i.e., software that records the keys the individuals press while typing) have been applied to the study of writing [22]. These tools offer a viable way to study students’ actions as they compose and edit their essays. One such tool, InputLog, has been developed to interface with NLP tools, which enables analyses that synthesize both keystroke and linguistic data.

Illustrative examples of the value of keystroke analyses stem from work on affect detection during writing [21; 23]. Writers’ affective states during writing—ranging from boredom and frustration to excitement and engagement—can have a significant impact on the writing experience and eventual products. However, these qualities may not be detectable from written products alone. How might keystroke patterns vary when writers are in a fluid, engaged “flow” state as compared to a frustrated struggle to generate ideas?

In recent work, Bisler and D’Mello (2013) [21] have begun to explore such questions. They collected individual difference measures and keystroke data from student writers to detect online affective states during writing (i.e., self-reported affective states in 15-second intervals). Their results indicated that a combination of behavioral (keystroke) measures and student-level indices was able to detect boredom, engagement, and neutral states between 11% and 38% above baseline. Similarly, Allen et al. (in press) [21] combined individual difference, linguistic, and keystroke indices to predict engagement and boredom across writing sessions. Their results suggested that these three categories of indices were successful in modeling students’ affective states during writing. Indices related to academic ability, text properties, and keystroke logs were able classify high and low engagement and boredom in writing sessions with 77% accuracy.

In sum, keystroke analyses hold the potential to reveal crucial data on students’ online writing experiences and processes that are normally invisible in product-based analyses alone.

1.2 Writing Pal

A long-term goal of our research is to improve personalized, adaptive learning and feedback within the Writing Pal (W-Pal) intelligent tutoring system [24]. W-Pal offers explicit strategy instruction, practice, and feedback for prompt-based persuasive essay writing for high school and early college students. Relative to other writing training systems (see [24] for a review), W-Pal is unique in its focus on explicit strategy instruction and its varied opportunities for practice (i.e., game-based strategy practice and essay writing practice). Strategy instruction is delivered via video presentations on canonical writing processes: prewriting, drafting, and revising. These videos feature virtual pedagogical agents who explain and demonstrate a variety of principles and strategies (see Figure 1 for a screenshot of the Freewriting Module). These lessons include: Freewriting and Planning (prewriting); Introduction Building, Body Building, and Conclusion Building (drafting); and Paraphrasing, Cohesion Building, and Revising (revising). After completing lessons, students unlock a suite of strategy practice mini-games. In these games, students reinforce their strategy knowledge through both generative and identification tasks. Game-based practice allows students to work on specific components of the writing process and strategies prior to applying them in a complete essay composition.

![Figure 1. Screenshot of the the Freewriting module](image)

1.2.1 W-Pal Essay Practice and Feedback

W-Pal also gives students the opportunity to practice writing persuasive essays and receive summative and formative feedback. Writing takes place in a word-processing interface where students can view the prompt, a “scratch-pad” for brainstorming and outlining, and the writing space. Once the essays are submitted, a combination of formative and summative feedback is provided. Like other AWEs, W-Pal employs NLP tools to extract linguistic data from essays, and implements a series of algorithms to assess quality and guide feedback delivery. In analyzing the text, the system considers characteristics across a variety of linguistic indices.

Summative feedback (see Figure 2) includes a holistic score on a 1-6 scale, with descriptors representing each level (i.e. “Great”). Formative feedback (see Figure 2) is given both at the essay-level (i.e. length, relevance, structure) and section-level (i.e. suggestions to improve an introduction). This formative feedback is designed to be specific, actionable, and aligned to strategies taught in the lessons. For example, students who submit essays with weak conclusions may receive feedback about summarizing key arguments from the body paragraphs in the conclusion. After viewing the feedback, students can revise their essays. In the
revision phase, essay feedback is displayed adjacent to the writing space, facilitating uptake of the recommendations. Previous research evaluating the efficacy of the W-Pal system has found that this training results in improved essay scores, increased strategy knowledge, and improved revising strategies [15; 25-26].

![Screenshot of the feedback window](image)

**Figure 2. Screenshot of the feedback window**

### 1.3 CURRENT STUDY

The current study investigates how and whether information about students’ writing behaviors within W-Pal can be recovered from basic behavioral data extracted from keystroke analyses. To this end, we calculate a number of indices based on the keystrokes pressed by students writers with the intent of modeling the quality and linguistic features of their essays. An overarching aim of this research is to develop online, stealth assessments of students’ writing processes that can inform new student models and system adaptivity. An increase in the sensitivity of W-Pal to students’ writing processes is expected to improve its ability to offer more nuanced and personalized feedback and recommendations.

We collected timed, persuasive essays written by undergraduate students and scored using the W-Pal algorithm [27]. Linguistic properties of the essays were assessed via Coh-Metrix [28] and WAT [29], which are automated NLP tools that calculates text information related to lexical, syntactic, cohesive, and rhetorical properties. In addition, we logged keystrokes during students’ writing session and calculated measures related to the general and temporal properties of these keystroke logs.

We hypothesized that these basic and time-sensitive keystroke indices would provide meaningful information about the writing processes enacted by students, which would subsequently relate to the quality and characteristics of their essays.

### 2. METHODS

#### 2.1 Participants

We recruited 131 undergraduate participants from a university in the United States, who received course credit. Students reported a mean age of 19.8 years, with 44.3% identifying as female, 64.1% Caucasian, 14.5% Asian, 7.6% African American, 7.6% Hispanic, and 6.1% as “Other.” Data for five students were lost due to computer error; thus, the final corpus comprised 126 essays.

#### 2.2 Data Collection Procedure

Participants wrote a timed (25-minute), prompt-based, persuasive essay. Essay prompts resembled typical SAT items, and students were not allowed to proceed until the full 25 minutes elapsed. Students typed their essays in the AWE component of W-Pal and all keystrokes were logged along with millisecond timestamps. Essays contained an average of 412.3 words ($SD = 159.9$, $min = 47.0$, $max = 980.0$).

### 2.3 Essay Scoring

Students’ essays were automatically scored using a computational algorithm that assigns scores on a scale from 1 (lowest) to 6 (highest). This algorithm relied on linguistic features computed by Coh-Metrix, the Writing Assessment Tool (WAT), and Linguistic Inquiry and Word Count (LIWC). For more details on this algorithm, see [27].

#### 2.4 Text Analyses

Linguistic properties of essays were assessed via two NLP tools: Coh-Metrix [28] and WAT [29]. These tools report hundreds of linguistic indices that relate to text structure, general readability, rhetorical patterns, lexical choices, and cohesion. For the current analyses, we selected four indices from Coh-Metrix and WAT that demonstrated theoretical ties to writing quality. We chose this limited number of indices to specifically examine whether and how the keystroke indices would map onto four key dimensions of the essays: lexical, syntactic, semantic, and cohesion.

**Word Frequency.** Coh-Metrix and WAT calculate multiple indices that describe the specific types of words used in texts. Word frequency measures, for instance, are used to assess how frequently certain words occur in the English language. Coh-Metrix reports indices of word frequency that are taken from the CELEX database. Additionally, Coh-Metrix reports the logarithm of word frequency for all words in a text. An index of log frequency is calculated because reading times are typically linearly related to the logarithm of word frequency rather than the raw word frequency [30]. For this reason, we chose to examine the log frequency of all words.

**Syntactic Complexity.** Additionally, Coh-Metrix and WAT contain a number of indices that describe the properties of the sentences in texts, such as the frequency of specific parts of speech and the complexity of their syntactic constructions. Sentence complexity is assessed by multiple indices. More complex syntax is typically associated with higher quality essays [28] and recent evidence suggests that working memory capacity is linked to the production of more complex syntax [31]. Here, we used the index mean number of words before the main verb as a proxy for sentence complexity.

**Semantic Diversity.** Semantic diversity refers to the number of unique concepts expressed in an essay. This measure is conceptually similar to measures of lexical diversity, but more strongly emphasizes the diversity of ideas rather than specific words. A semantic diversity score is calculated in WAT using Latent Semantic Analysis (LSA) [32] and is operationalized as the ratio of semantically independent concepts to the total number of word types in an essay.

**Global Semantic Cohesion.** Global semantic cohesion is also calculated in WAT using LSA. Here, we used the index LSA (start-to-end), which calculates the degree to which the introduction and conclusion of an essay contain semantically similar information. We chose this index (rather than examining the semantic similarity between all the paragraphs) because higher-quality essays typically share semantic content in the opening and closing paragraphs, but bring in outside information in the form of arguments and evidence in the body paragraphs.
3. KEystroke Analyses

To investigate whether and how students’ writing behaviors were related to the quality and linguistic properties of their essays, we computed a number of keystroke indices. In particular, we calculated both basic keystroke indices (i.e., indices that were aggregated across the entire essay), as well as time-sensitive keystroke indices (i.e., indices that accounted for the temporal nature of the keystroke data).

3.1 Basic Keystroke Indices

Basic keystroke indices aggregated the number of specific writing events (e.g., pauses and backspaces) that occurred across an entire writing session. These basic indices are deliberate replications of indices from previous studies because they have been successfully used to model students’ affect during writing [21; 23]. Table 1 provides an overview of these indices.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verbosity</td>
<td>Number of keystrokes per essay</td>
</tr>
<tr>
<td>Backspaces</td>
<td>Number of backspaces per essay</td>
</tr>
<tr>
<td>Largest Latency</td>
<td>Largest time difference between keystrokes during essay writing</td>
</tr>
<tr>
<td>Smallest Latency</td>
<td>Smallest time difference between keystrokes during essay writing</td>
</tr>
<tr>
<td>Median Latency</td>
<td>Median of all the differences in time between keystrokes per essay (not including initial pause)</td>
</tr>
<tr>
<td>Initial Pause</td>
<td>Length of the first pause of an essay writing session</td>
</tr>
<tr>
<td>0.5 Second Pauses</td>
<td>Number of pauses above .5 seconds and below 1 second</td>
</tr>
<tr>
<td>1 Second Pauses</td>
<td>Number of pauses above 1 second and below 1.5 seconds</td>
</tr>
<tr>
<td>1.5 Second Pauses</td>
<td>Number of pauses above 1.5 seconds and below 2 seconds</td>
</tr>
<tr>
<td>2 Second Pauses</td>
<td>Number of pauses above 2 seconds and below 3 seconds</td>
</tr>
<tr>
<td>3 Second Pauses</td>
<td>Number of pauses above 3 seconds</td>
</tr>
</tbody>
</table>

3.2 Time-Sensitive Keystroke Indices

Despite the importance of basic keystroke indices, indices that aggregate behavioral patterns over the course of an entire essay session can miss out on important temporal variability. For instance, consider the time series depicted in Figure 3. This plot shows the number of keystrokes pressed by one student writer within each 30 second window of a writing session. The student clearly did not maintain stable behavioral patterns throughout the writing session; instead, she engaged in periods of high and low activity. Analyses that are restricted to basic indices necessarily ignore this variability. We hypothesize that investigations into the temporal structure of the keystrokes (i.e., the distributions of events in time) will provide meaningful information about students’ writing processes beyond the basic aggregated measures.

![Figure 3. Variability of keystroke patterns for a single student](image_url)

Table 2. Time-Sensitive Keystroke Indices

<table>
<thead>
<tr>
<th>Description</th>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard deviation of the number of events in each time window</td>
<td>StDev Events</td>
</tr>
<tr>
<td>Slope of the linear regression applied on the time series</td>
<td>Slope Degree</td>
</tr>
<tr>
<td>Shannon’s Entropy calculated for the number of events in the windows normalized by the total number of events for the overall time series</td>
<td>Entropy</td>
</tr>
<tr>
<td>Uniformity of the time series (Jensen-Shannon divergence method), which is a symmetric and bounded function of similarity that calculates the similarity between two distributions: a uniform probability distribution of 1/n (i.e., a constant typing rate) and the probability of key presses in a given window (i.e., the actual time series produced by the student)</td>
<td>Degree of Uniformity</td>
</tr>
<tr>
<td>Number of time windows for which the direction of the evolution of keystroke events changes. This reflects inconsistency in writing rates across the windows</td>
<td>Local Extremes</td>
</tr>
<tr>
<td>Average recurrence of events across the time windows. This recurrence is expressed as the distances between time windows that contain at least one keystroke event. This measure is useful for identifying writing pauses. If each time window has at least one event, recurrence is 0, whereas if students take long pauses that occasionally result in time windows of 0 events, recurrence increases (if they write every two time windows, recurrence will be one).</td>
<td>Average Recurrence</td>
</tr>
<tr>
<td>Standard deviation of the recurrence across the time windows</td>
<td>StdDev Recurrence</td>
</tr>
</tbody>
</table>

Note: All time-sensitive keystroke indices were calculated using 30- and 60-second time windows.

To this end, we calculated a number of new indices that we have classified as time-sensitive keystroke indices. These indices deliberately take the within-subject temporal distribution of keystroke events into account. The time series of keystrokes...
generated during students’ sessions were first separated into non-overlapping windows of 30 and 60 seconds to account for variability across different scales. These individual windows contained information about the number of keystroke events that occurred in each time window. The time-sensitive keystroke indices were then separately generated based on each of the two window intervals (see Table 2).

3.3 Statistical Analyses
Statistical analyses investigated whether basic and time-sensitive keystroke indices accounted for variability in student writing performance. Pearson correlations were first calculated between the holistic essay scores and the keystroke indices obtained from the writing sessions (see Tables 1 and 2). Indices that displayed a significant or marginally significant correlation with essay scores \( p < .10 \) were retained in the analysis.

Normality of the indices was assessed with skew, kurtosis, and visual data inspections, and no indices were removed based on these inspections. Range transformations \((0-1)\) were applied to ensure that the keystroke and linguistic indices were on the same scale. Multicollinearity was then assessed among the indices \( r > .90 \). When two or more indices demonstrated multicollinearity, the index that correlated most strongly with essay scores was retained in the analysis.

A linear regression analysis\(^1\) was conducted using M5-prime feature selection to assess which of the remaining keystroke indices were most predictive of essay scores. To avoid overfitting the model, we chose a ratio of 15 essays to 1 predictor, which allowed for a maximum of eight indices to be entered into the model, given that there were 126 essays included in the analysis.

We first conducted the regression analysis on the entire corpus, and then validated the model using ten-fold cross-validation with shuffled sampling. In this cross validation analysis, the corpus was first split into 10 “folds” and each fold was individually removed from the corpus for each analysis and the remaining essays were used as the training set. We tested the accuracy of the linear regression model by examining its ability to model the omitted fold. The process was repeated until each fold was omitted once in the test set. This analysis therefore allowed us to test the model’s accuracy on independent sets of data \((i.e.,\ data\ that\ are\ not\ in\ the\ training\ set)\). If the overall model and the model generated by the cross-validation analysis are similar, our confidence in model stability is increased.

Following this essay score analysis, similar follow-up analyses were conducted using the keystroke indices to predict the linguistic features of the essays. For these analyses, we followed the same procedure detailed above.

4. RESULTS
4.1 Key strokes and Essay Quality
Pearson correlations were calculated between the basic and time-sensitive keystroke indices and students’ holistic essay scores to examine the strength of the relationships among the variables. The correlation analysis revealed that there were 10 keystroke indices that demonstrated a significant relation with holistic essay scores and did not demonstrate multicollinearity with each other. To avoid overfitting the model, we only selected the eight indices that were most strongly correlated with essay scores. These eight indices are listed in Table 3.

| Table 3. Correlations between Essay Scores and Keystroke Indices |
|-----------------|-----|------|
| Keystroke Index    |   r |  p   |
| Verbosity          | 0.819| <.001|
| Local Extremes (30s time window) | -0.476| <.001|
| Entropy (30s time window) | 0.472| <.001|
| Median Latency     | -0.436| <.001|
| StdDev Events (30s time window) | 0.397| <.001|
| Largest Latency    | -0.359| <.001|
| Backspaces         | 0.308| <.001|
| StdDev Recurrence (30s time window) | -0.297| =.001|

A linear regression analysis was calculated with the eight keystroke indices as predictors of students’ essay scores \( (score\ range: 1-6)\). This analysis yielded a significant model, \( R^2 = .758\), \( RMSE = 0.377\), \( p < .001\), with three variables that combined to account for 76% of the variance in the essay scores: Verbosity \( [β = 1.03, p < .001]\), Largest Latency \( [β = .99, p < .001]\), and Backspaces \( [β = .39, p < .001]\). The follow-up ten-fold cross validation analysis produced a significant model with similar statistics, \( R^2 = .737, RMSE = 0.386\).

An interesting question is whether additional indices provided useful information about the essay quality once Verbosity was removed from the analysis. That is, including the total number of key presses may suppress the important role of other writing behaviors. We conducted a second regression analysis that excluded Verbosity. This regression yielded a significant model, \( r = .778, R^2 = .606, RMSE = 0.482, p < .001\). Six variables were significant or marginally significant predictors in the regression analysis and combined to account for 61% of the variance in students’ essay scores: StdDev Events (30s) \( [β = 0.529, p < .001]\), Entropy (30s) \( [β = 1.047, p < .001]\), StdDev Recurrence (30s) \( [β = -0.509, p < .001]\), Backspaces \( [β = 0.209, p < .001]\), Local Extremes (30s) \( [β = -0.176, p < .05]\), and Median Latency \( [β = -0.141, p = .096]\). As above, the cross validation model produced similar results, \( R^2 = .588, RMSE = 0.534\).

In sum, these correlation and regression analyses indicate that better writers pressed more keys (both characters and backspace) over the course of their writing session. They also maintained a more consistent rate across the 30 second time windows \((i.e.,\ whether\ they\ typed\ or\ not\ within\ the\ individual\ time\ windows)\), as measured by Entropy, Local Extremes, and StdDev Recurrence indices, but exhibited greater variability in the number of keystroke events within the 30s time windows (StdDev Events). Additionally, these students’ keystroke logs were characterized by shorter pause times as measured both by the Median and Largest Latency indices. Taken together, these findings demonstrate that writing fluency—the ease and consistency with which writers generate text—is a key indicator of proficiency \((e.g.,\ [33])\). This work both confirms and extends prior research by investigating a

---

\(^1\) We investigated the usefulness of a number of regression and neural net techniques in the current study. However, due to space limitations, these models are not reported. In the end, we report the the linear regression models because this approach yielded the strongest and most stable models.
feature of higher quality writing using process analyses rather than post-hoc linguistic analyses alone.

4.2 Keystrokes and Linguistic Features

Our second aim was to investigate whether keystroke indices were related to specific linguistic features of the essays. Pearson correlations were calculated between the keystroke indices and the four linguistic variables calculated by Coh-Metrix and WAT. These analyses were then followed by a regression analysis, and validated using ten-fold cross validation. The statistical information for these resulting models is provided below.

Word Frequency. The word frequency regression analysis yielded a significant model, $R^2 = .185$, $RMSE = .179$, $p < .001$. Three variables were significant or marginally significant predictors: 2 Second Pauses $[\beta = -.278, p < .01]$, Initial Pause $[\beta = .203, p < .05]$, and 0.5 Second Pauses $[\beta = .208, p = .06]$. The cross validation model was significant, $R^2 = .204$, $RMSE = .187$.

Syntactic Complexity. None of the keystroke indices were significantly or marginally significantly correlated with the selected measure of syntactic complexity.

Semantic Diversity. The analysis to predict the semantic diversity in essays yielded a significant model, $R^2 = .375$, $RMSE = .123$, $p < .001$. Five variables were significant predictors in this regression analysis: 1 Second Pauses $[\beta = -.379, p < .001]$, StdDev Events (30s) $[\beta = -.361, p < .01]$, Slope Degree (30s) $[\beta = .336, p < .01]$, Median Latency $[\beta = .265, p < .05]$, and Local Extremes (60s) $[\beta = .173, p < .05]$. The cross-validation analysis yielded a significant model, $R^2 = .255$, $RMSE = .133$.

Global Semantic Cohesion. Analyses to predict global semantic cohesion based on keystroke data yielded a significant model, $R^2 = .194$, $RMSE = .238$, $p < .001$ with four significant predictors: StdDev Events (30s) $[\beta = .477, p < .01]$, 3 Second Pauses $[\beta = .424, p < .001]$, Verbosity $[\beta = .337, p < .01]$, and Median Latency $[\beta = .307, p < .05]$. The model produced by the cross-validation analysis was significant, $R^2 = .160$, $RMSE = .244$.

The results of the linguistic analyses indicate that the basic and time-sensitive keystroke indices were meaningfully related to the linguistic features of students’ essays at multiple levels. Notably, however, the linguistic regression models were weaker than the essay score model, and the findings were less robust to the cross-validation procedure.

The model generated to predict semantic diversity was the strongest of the linguistic models. This analysis indicated that more semantically diverse essays were related to shorter pauses, with more variability at the 60-second time window (Local Extremes), but less variability at the 30-second time windows. The global semantic cohesion and word familiarity models were also significant with keystroke indices for both accounting for just under 20% of the variance in the linguistic properties. Finally, the syntactic complexity measure was not significantly related to any of the keystroke indices, indicating that perhaps behavioral patterns do not manifest in the different sentence structures produced by writers.

5. DISCUSSION

AWE systems provide an environment for students to receive writing instruction and engage in deliberate practice with summative and formative feedback [12]. Despite the general success of their scoring algorithms (e.g., [13-14; 27]), however, the pedagogical elements of these systems have much room for improvement. For instance, one major weakness of AWE systems is that they typically only adapt to student users based on individual essay drafts. System developers tend to rely on NLP methods to examine the quality of students’ written products; yet, information about their behavioral processes is largely ignored.

In the current study, we used system logs of keystrokes to develop online assessments of students’ writing performance. The behavioral processes enacted by writers are important elements of writing skill [1; 22]; therefore, our aim was to determine whether we could assess and model the quality and linguistic properties of students’ essays by calculating indices related to their typing behaviors. Basic and time-sensitive keystroke indices were calculated to analyze the behavioral patterns enacted by student writers. These indices provided information about writing processes at both the aggregate level (e.g., total number of pauses and backspaces) as well as information about how these behaviors unfolded over time. The results revealed that keystroke indices were able to model over three-quarters of the variance in students’ essay scores. Additionally, these indices were able to model the linguistic properties of the essays at multiple levels.

The essay score analyses revealed that 10 keystroke indices were significantly correlated with students’ holistic essay scores. This is important because it indicates that information about the quality of students’ essays can be detected by analyzing their behavioral processes. Further, the two regression analyses revealed that the total number of keystrokes pressed by writers provided the most predictive power in the model, but that without this measure of Verbosity, the remaining indices were still about to account for 61% of the variance in essay scores.

These initial analyses of essay score indicate that fluency may be an important skill that is captured by the keystrokes. In our study, the students who produced higher-quality essays were also more consistent in their typing (i.e., whether they typed or not) across the 30 second time windows, yet they had higher variability in the number of keystroke events they produced in these time windows. This finding suggests that these students’ writing sessions may have been characterized by short (rather than long) patterns of writing and pausing. Some confirmation for this intuition is found in the the negative correlations between essay score and pause times (i.e., Median and Largest Latency). However, future research will need to examine these writing-pause patterns more closely. It may be the case, for instance, that short pauses are indicative of thoughtful writing, such as the search for appropriate words or phrases rather than “freewriting” behavior. Long pauses, on the other hand, may be indicative of mind wandering that warrants system intervention.

Follow-up linguistic analyses similarly revealed important information about the role of behavioral processes in writing. These analyses first indicated that the basic and time-sensitive keystroke indices were significantly related to the linguistic features of students’ essays at the lexical, semantic and global cohesion levels, but not at the syntactic level. This indicates that keystroke indices may be picking up on specific meaning-making processes, rather than differences in cognitive factors, such as working memory capacity. For instance, semantic diversity represents the number of semantically related concepts that appear in students’ essays, which may map onto the differences in the content that students chose to include in their essays. Syntactic complexity, on the other hand, is much more weakly related to the
meaning of a particular text and, instead, may be indicative of individual differences in specific cognitive skills (e.g., [31; 34]).

It is important to note that the keystroke indices accounted for a smaller amount of the variance in linguistic properties than in the overall essay scores. This suggests that variations in students’ behavioral patterns may manifest in the properties of students’ essays in different ways depending on the specific context. For instance, long pauses may be more indicative of cohesion if students are writing about an unfamiliar topic that requires more deliberate planning. On the other hand, if students are writing in response to a familiar or emotionally charged topic, it may be the case that essay cohesion will be associated with rapid typing with minimal pauses. The results of these follow-up analyses suggest that future analyses may need to use content-based information to make predictions about the relevance and interpretation of particular keystroke indices. Analytic techniques that allow the system to take past behavior and prompt content into consideration, for instance, could go a long way in improving the interpretability of these patterns.

These results are promising and suggest that keystroke indices can be utilized to uncover important information about the behavior and performance of student writers. Here, we analyzed the keystrokes produced for a short, prompt-based essay task. In the future, additional studies will be conducted to specifically examine how these keystroke patterns map onto writing across different genres, contexts, and difficulty levels. For example, multiple writing sessions could be collected for each participant, with prompt difficulty, genre, or audience varying across these sessions. This research design would help to disentangle signals that vary across multiple factors, such as boredom and difficulty. Another area for future research lies in the calculation of more sophisticated keystroke indices, as well as the integration of keystroke indices with other system information. We used only keystroke indices as our predictors because we were interested in the degree to which simple behavioral measures alone could predict information about students’ essays. In future studies, it will be important to consider additional indices that may be related to the context of these writing behaviors. For instance, if we aim to model students’ engagement during writing, it will be important to collect additional information from our systems, such as their prior writing behaviors (e.g., on previous essays, or from original to revised drafts), as well as the linguistic content of the essays.

The overarching goal of this research is to enhance AWE systems such that they provide feedback and instruction that is more attuned to writers’ processes. Eventually, we aim to be able to identify specific behavioral patterns associated with different writing processes, which will allow us to provide students with more pointed, online feedback and instruction. For example, through the combination of multiple keystroke indices, systems may be able to distinguish when students are experiencing writer’s block as opposed to when they are engaged in the task, but have paused to think. If writer’s block were detected, W-Pal could then ask students if they need help or offer specific strategies and practice opportunities for idea generation.

Overall, our results suggest that time-sensitive behavioral data can (and, in our opinion, should!) be used to help drive more personalized feedback and instruction in computer-based learning environments. Although a number of future studies are needed to investigate how this keystroke information can be used most effectively, the current study takes a strong first step in revealing the power of these indices.

6. ACKNOWLEDGEMENTS

The research reported here was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305A120707 to Arizona State University. The opinions expressed are those of the authors and do not represent views of the Institute or the U.S. Department of Education.

7. REFERENCES


Automatic Gaze-Based Detection of Mind Wandering during Narrative Film Comprehension

Caitlin Mills, Robert Bixler, Xinyi Wang, & Sidney K. D’Mello
University of Notre Dame
384 Fitzpatrick Hall, Notre Dame, IN, 46556, USA
{cmills4, rbixler, xwang24, sdmello}@nd.edu

ABSTRACT
Mind wandering (MW) reflects a shift in attention from task-related to task-unrelated thoughts. It is negatively related to performance across a range of tasks, suggesting the importance of detecting and responding to MW in real-time. Currently, there is a paucity of research on MW detection in contexts other than reading. We addressed this gap by using eye gaze to automatically detect MW during narrative film comprehension, an activity that is used across a range of learning environments. In the current study, students self-reported MW as they watched a 32.5-minute commercial film. Students’ eye gaze was recorded with an eye tracker. Supervised machine learning models were used to detect MW using global (content-independent), local (content-dependent), and combined global+local features. We achieved a student-independent score (MW F_1) of .45, which reflected a 29% improvement over a chance baseline. Models built using local features were more accurate than the global and combined models. An analysis of diagnostic features revealed that MW primarily manifested as a breakdown in attentional synchrony between eye gaze and visually salient areas of the screen. We consider limitations, applications, and refinements of the MW detector.

Keywords
mind wandering; film comprehension; machine learning; eye gaze

1. INTRODUCTION
Mind wandering (MW) reflects an attentional shift from task-related to task-unrelated thoughts [31]. MW is estimated to consume half of our everyday thoughts [19] and can occur at almost any time – driving down the road, eating a meal, or during a classroom lecture. There are some benefits to our innate ability to MW, specifically with respect to planning and creativity [34]. However, MW has some detrimental effects as well, particularly in the realm of education [30]. A recent meta-analysis across 88 independent samples indicated that MW was negatively correlated with performance, and that the negative relationship was stronger for more complex tasks such as reading comprehension [26]. Given the negative impact of MW on learning [29, 30], it is important to develop attention-aware systems that can reorient attention when MW occurs [8]. However, these systems require reliable MW detection, which is the focus of this work.

MW detection can be particularly challenging since MW is an internal state with few overt markers (unlike some emotions per se). It can even be difficult for people to realize when they are MW, as it can occur without metacognitive awareness [30]. Moreover, the onset and duration of MW cannot be clearly demarcated as with other disengaged behaviors, such as gaming the system or WTF (Without Thinking Fastidiously) behaviors [1, 25].

In the present study, we focus on detecting MW in the novel educational context of narrative film comprehension – a more complex task than self-paced reading where most MW detection efforts have focused on. We chose this task for two reasons. First, a large number of students from all over the world watch educationally relevant films and recorded lectures daily, particularly in the advent of massive open online courses (MOOCs). Second, MW is quite frequent in online video lectures: students report MW around 40% of the time while viewing lectures [29, 33], so there is considerable promise to detecting and responding to MW in this context.

1.1 Background and Related Work
Only one study (to our knowledge) has attempted MW detection while students viewed dynamic visual scenes, such as the narrative film we consider here. Pham and Wang [25] detected MW while students watched video lectures on a smart phone with a MOOC-like application and responded yes or no to thought probes during the lectures. They used student heart rate (extracted via photoplethysmography) to train classifiers to detect MW. They achieved a 22% greater than chance detection accuracy, thereby providing some initial evidence that MW detection is feasible in this context.

Aside from [25], other MW detection efforts have been limited to self-paced reading. In one of the first MW detection studies [10], students read aloud and then paraphrased biology paragraphs. They were periodically asked to report zone outs during reading on a 1 (all the time) to 7 (not at all) scale. Supervised machine learning models trained on acoustic-prosodic features to classify between “high” (1-3 on the scale) versus “low” zone outs (5-7 on the scale) achieved a 64% accuracy. However, this study did not adopt a student-independent validation approach, so it is unclear how well their detector would generalize to new students.

Other research has utilized log-file information to detect MW during self-paced reading. In one study [23], MW reports were collected via pseudo-random thought probes during self-paced computerized reading. Students responded either “yes” or “no” about whether they were MW at the time of the probe. Using textual features and reading behaviors from log-files, supervised machine learning models were able to detect MW with a 21% above-chance accuracy. Similarly, [12] attempted to predict MW during reading using textual features (e.g., difficulty, familiarity, and reading time), but it is not clear if their method, which utilized researcher-pre-defined thresholds, would generalize more broadly.

Denotes equal contribution by authors.

Proceedings of the 9th International Conference on Educational Data Mining
Researchers have also adopted sensor-based approaches for MW detection during reading. Blanchard et al. [4] used an Affectiva Q sensor to record both galvanic skin response and skin temperature while participants read texts on research methods and periodically provided MW reports in response to thought probes. Their models attained a kappa value of .22 using a combination of peripheral physiology and contextual features (e.g., page numbers).

Eye gaze is perhaps one of the most promising modalities for MW detection due to the so called eye-mind link [27], which posits a coupling between eye movements and attentional focus. Several studies have thus built MW detectors using eye gaze features. The first study collected data from 84 students during self-paced reading of four texts on research methods [7]. MW reports were collected in response to thought probes triggered when gaze was fixated on predefined words on the screen. Supervised classification models were built from 27 gaze features and validated in a student-independent fashion. The authors achieved an accuracy of 60% after downsampling the data. Since downsampling was applied to both the training and test sets, it is unclear how the models would perform when presented with data that reflected the original skewed class distributions.

Their work was extended using a larger dataset of 178 students from two different universities and a wider array of 80 features, including blink and pupil features [2]. Students also read four texts on research methods, and MW reports were collected in response to nine pseudorandom probes that occurred between four to twelve seconds from the beginning of a page of text. Supervised models were built using an extended feature set and were cross-validated in a student-independent fashion. The models achieved an accuracy of 72% (31% above chance) when validated with a test set that maintained the original class distributions. Further, in [2], the authors provided evidence for the predictive validity of the model by showing that it predicted posttest scores at rates higher than self-reported MW, even after controlling for prior knowledge.

The results from this study indicate that MW can be detected from eye gaze during self-paced reading with moderate accuracy. However, there is an open question about the use of eye gaze to detect MW in additional contexts— in particular, for more complex stimuli like dynamic visual scenes. One study [35] provided evidence that eye movements can be predictive of attention while viewing short video clips. In this study, participants watched video clips in two different conditions: (1) without any distractions (attending) and (2) while performing a mental calculation (not attending). Results indicated that eye movements toward pre-determined salient locations in the scene could identify the watching condition (attending vs. not attending) with a 80.6% accuracy, albeit this is not quite the same as MW detection.

We should note that there is still some debate whether eye movements can be driven by salient features of the stimulus (exogenous control) or through conscious control (referred to as endogenous control). There is some research to suggest that eye movements are primarily driven by exogenous control. For example, previous research has shown that different viewers tend to fixate on the same locations [24], a phenomenon known as attentional synchronicity, which suggests exogenous control. However, other research pointed out that interesting objects are often the most visually salient [11]. Thus, it is possible that viewers fixate on the same locations because of top-down processes (endogenous control), as opposed to simply looking at what is salient. Additional evidence for endogenous control comes from a study which found that task instructions can have an effect on eye movements while viewing dynamic visual scenes [32]. The researchers found that participants looked at more peripheral and less visually salient areas of the scene when instructed in order to determine where the visual scenes were derived from compared to a general viewing task. Thus, eye movements related to endogenous control might be particularly revealing about MW.

The current study utilizes this idea to compute features that capitalize on the relationship between eye movements and visually salient regions in the film.

1.2 Current Study and Novelty

In this paper we present one of the first attempts to automatically detect MW during narrative film viewing in a manner that generalizes to new students. We leverage what has been learned in previous work using eye gaze to detect MW during reading, while also developing theoretically-grounded features to improve detection accuracy in this novel context.

MW detection during film viewing poses unique challenges compared to reading, which has been the most common context for MW detection thus far. For one, eye movements are much more predictable during reading since the words on the screen are static. In addition, reading consists of fixations (periods where the gaze position is relatively stable) and saccades (rapid movements between fixations), while the dynamic nature of film also yields smooth pursuits (eye movements that follow a moving stimulus).

Second, the film played continuously without any clear breaks, presenting an additional challenge for MW detection. This is in contrast to reading tasks, which are segmented by page breaks. Thus, a novel method was devised to segment eye gaze data into instances for classification.

Finally, the dynamic nature of film allowed for novel content-dependent features that can be computed from dynamic areas of interest (AOI). Unlike reading, AOIs are particularly meaningful in a film viewing context because of the distinctive visual content areas that dynamically change throughout a film. In this study, AOIs were computed from both plot-related and visually salient regions.

2. DATA COLLECTION

This study utilized a subset of data reported by Kopp el. [21].

2.1 Participants

Eye gaze data was collected for 60 undergraduate students from a private Midwestern university. Students were 20.1 years old on average and 66% of the students were female.

2.2 Materials

Students watched “The Red Balloon,” a 32.5 minute French film with few English subtitles (9 in all). The film was displayed on a computer screen with a resolution of 1920 x 1080. The film depicts the story of a young boy and a red balloon that follows him and can inexplicably move on its own. This film was chosen because it is unlikely that many students had previously seen it, which could have affected their propensity to mind wander. The film has also been used in previous film comprehension studies [36].

All data were collected using a Tobii TX 300 eye tracker that was attached to the bottom of the monitor. Eye gaze was recorded with a sampling frequency of 120 Hz for the first 14 participants (due to experimenter error), after which the sampling frequency was adjusted to 300 Hz. This difference was taken into account when filtering the gaze data as discussed below.
2.3 Mind Wandering Reports

Students were asked to self-report MW while they watched the film by pressing labeled keys on a standard keyboard. A short beep sounded to register their response, but the film was not otherwise interrupted. A self-caught MW report method was chosen as opposed to a probe-caught report method (where students are probed to report MW at pseudo-random intervals) in order to minimize disruption, which was critical as the film played without interruption.

Students were asked to differentiate between two different types of MW using separate keys: either task-unrelated thoughts (thoughts completely unrelated to the film such as upcoming vacation plans) or task-related interferences (thoughts related to the task but not the content of the film, such as “This film is boring”). For the present analyses, both task-unrelated thoughts and task-related interference were grouped as MW. There was a total of 616 MW reports. On average, students reported 10.3 instances of MW during the film ($SD = 7.9$; $Min = 1; Max = 31$).

2.4 Procedure

Students were asked to sit comfortably at a desk in front of the monitor before beginning the eye-tracker calibration process. There were no restrictions on head movements, making the film viewing experience more ecologically valid than if a headrest was used. Students were randomly assigned to one of two conditions before the film started: in one condition, they read a short story explaining the movie plot [22] while students in the second condition read an unrelated baseball-themed story [1]. The experimental manipulations were part of a larger study and are not used here (more details can be found in [21]). Finally, students were given instructions for how to report MW and then the film began. Students completed a multiple choice comprehension assessment after viewing the film, but this data is not analyzed here.

3. MODEL BUILDING

3.1 Eye Movement Detection

Eye gaze was converted to eye movements (fixations, saccades, smooth pursuits, etc.) in order to filter out some of the inherent noise in raw eye gaze data. We first averaged the raw data from the right and left eyes. A simple moving average filter was then applied to the gaze points in order to smooth the signal while retaining the same sampling frequency. The filter used a window size of five samples for the 120 Hz data and seven samples for the 300 Hz data.

Eye movements were detected using a velocity based algorithm [18, 20]. These algorithms generally use thresholds to classify gaze points as fixations, saccades, or smooth pursuits. The algorithm first classified gaze points with a velocity greater than 110 degrees of visual angle/s as saccades. It then classified gaze points with a velocity lower than five degrees of visual angles as fixations. Any remaining gaze points were classified as smooth pursuits. The visual angle thresholds used were based on previous research [17].

3.2 Film Segmentation

Next, we segmented the continuous stream of eye gaze data into MW and non-MW segments. Each segment had three components: gap, window, and offset (see Figure 1). The gap was the number of seconds between adjacent segments and could be adjusted to change the ratio of MW to non-MW segments. The window was the portion of the segment used to compute features. The offset was the number of seconds between the MW report (the moment when the student pressed the key on the keyboard) and the end of the window. An offset was used in order to discard data affected by the student’s motion to press the key when reporting MW. An offset size of three seconds was deemed appropriate based on observation of recorded videos.

The process began by creating a MW segment prior to each MW report (segment 2 in Figure 1). The data prior to the MW segment were then considered to be non-MW segments (segment 1) after accounting for the gap. There was no offset for non-MW segments as no key presses were involved.

![Figure 1. Hypothetical example of segmented data](image)

There were several considerations when choosing the window and gap sizes. The segment size (sum of the window, offset, and gap sizes) determined both the number of available instances (segments) and the MW rate as shown in Table 1. Models were built with segment sizes of 45, 55, and 65 seconds, resulting in MW rates that ranged from .256 to .323 and number of instances from 2401 to 1626, thereby allowing us to explore how these two factors affected classification accuracy. For each of these segment sizes, the window size was also varied. In all, we considered window sizes of 10, 15, 20, and 25 seconds.

| Table 1. Effect of segment size on number of segments and MW rate |
|---|---|---|
| Seg. Size (secs) | Number of Segs. | MW Rate |
| 45 | 2401 | .256 |
| 55 | 1931 | .297 |
| 65 | 1626 | .323 |

3.3 Feature Engineering

A total of 143 features were computed from the window in each segment. We considered global features, which were independent of the film content, and local features, which were content specific.

3.3.1 Global Features

There were 88 total global features. Of these, 75 were computed from measures of the eye movements, including fixations, saccades, and smooth pursuits, as well as blinks and pupil diameter. Fixation features were computed from the fixation durations (ms). Saccade features were computed from the saccade durations (ms), amplitudes (degrees of visual angle), velocities (degrees of visual angle/s), relative angle (degrees of visual angle between two consecutive saccades), and absolute angle (degrees of visual angle between a saccade and the x-axis). Smooth pursuit features were computed from the duration (ms), length (degrees of visual angle), and velocity (degrees of visual angle/s) of smooth pursuits. The following descriptive statistics of the distributions were used as the features: minimum, maximum, mean, median,
standard deviation, skew, kurtosis, and range. Counts of each eye movement type were also included as features.

Eight global features were obtained from pupil diameters, which were first z-score standardized at the student-level. The minimum, maximum, median, standard deviation, skew, kurtosis, and range were computed for the standardized pupil diameter distributions from each window and used as features.

There were five additional global features: blink count, mean blink duration, the ratio of total fixation duration to total saccade duration, the proportion of horizontal saccades, and the fixation dispersion.

3.3.2 Local Features

We identified two types of areas of interest (AOIs), Red Balloon AOIs and Visual Saliency AOIs, and computed features based on the locations of the AOIs in each frame. Red Balloon AOIs were used because the red balloon is one of the main objects in the film and endogenous attentional control might direct students to focus on these AOIs despite competing content. OpenCV [4], an open source computer vision software library, was used to isolate the red balloon from the rest of the image using a red color mask. A bounding box was drawn around a contour of the resultant image for each frame in which the balloon appeared (as shown on the left in Figure 2). Local features related to the red balloon were only computed for frames where it was present (58.2% of frames).

We manually examined each frame to ensure that the AOIs were computed correctly. The red balloon was present in 27,262 out of the 46,851 frames. An AOI was constructed for 26,925 of those frames, yielding an accuracy of 98.7%. The frames where the red balloon was missed could be attributed to lighting conditions (making the red balloon appear darker and thus difficult to distinguish from other parts of the scene), the small size of the red balloon, or the majority of the red balloon being off screen or occluded. These frames were left untouched. An additional 8 frames incorrectly had an AOI around an object that was not the red balloon. The AOI was simply deleted from these frames.

Visual Saliency AOIs were used because visually salient areas are known to attract eye gaze [11]. Although, the visual saliency and red balloon AOIs overlap in some cases, as in Figure 2, the visual saliency AOI can be computed for frames without the red balloon. The MATLAB implementation of a Graph-Based Visual Saliency algorithm [16] was used to produce a visual saliency map for each frame based on color, intensity, orientation, contrast, and movement. An area of no more than 2,000 pixels (1.1% of the screen area) surrounding the most salient point were retained and the remaining pixels were set to an intensity of 0. Similar to above, a bounding box was drawn around the largest contour of the processed image.

Local features were computed based on the relationship between the AOIs and each type of eye movement. The features included: (1) AOI distance, (2) AOI intersection, and (3) saccade landing. There were 32 AOI distance features, which captured the distance between the AOI and gaze positions. AOI distance features were computed as the distance between each fixation point or smooth pursuit point and the center of the AOI for each frame in the window. Fixation points were generated for each frame at the centroid of the fixation. Smooth pursuit points were generated for each frame using linear interpolation from the onset to the offset of each smooth pursuit. The minimum, maximum, mean, median, standard deviation, skew, kurtosis, and range of the measured distances were then computed for each eye movement, resulting in 16 features for each type of AOI (32 in all).

There were 12 additional AOI intersection features. These were calculated as the proportion of frames in which a fixation or smooth pursuit point was within the AOI bounding box. Four of these features used the original dimensions of the AOI bounding box. An additional eight used a bounding box expanded by either one or two degrees of visual angle in order to account for inaccurate eye gaze or cases where the AOI was small in size.

Finally, there were 12 saccade landing features. For each AOI, there was a single feature that captured the number of saccades onto, away from, or within the AOI bounding box, which resulted in six features (3 per AOI). An additional six features were computed using a bounding box expanded by one degree of the visual angle to accommodate gaze tracking errors or small AOIs.

In all, there were 56 local features (32 AOI distance, 12 AOI intersection, and 12 saccade landing).

3.4 Model Building

Twelve supervised machine learning algorithms from Weka [14] were used to build models that discriminated between MW and non-MW instances (windows). The following classifiers were used: Bayes network; naïve Bayes; logistic regression; SVM; k-nearest neighbors; decision table; Rip; C4.5 decision tree; random forest; random tree; REPTree; and REPTree with bagging.

We also varied four external parameters: (1) feature type; (2) window and segment size; (3) feature selection percentage; and (4) sampling method. With respect to feature type, models were built with global features, local features, or both global and local features using feature-level fusion.

The segment and window size(s) were varied because there are various tradeoffs at play. Specifically, a larger segment size resulted in fewer instances but a higher MW rate, thereby reducing class imbalance. A larger window size afforded more data for each instance, but it also reduced the number of instances available for segments with the same gap size (e.g., a window size of 30 and gap size of 15 resulted in fewer instances than a window size of 40 and gap size of 15). Thus, models were built with segment sizes of 45, 55, or 65 seconds, and window sizes of either 10, 15, 20, or 25 seconds.

Feature selection was used on the training set of each cross-validation fold (see below). Features were ranked using correlation-based feature selection (CFS) [15] from Weka and the top 30%, 50%, or 80% of features ranked were retained.

Class imbalance poses a well-known challenge for supervised classifiers. Hence, training sets were resampled using...
downsampling or oversampling. Downsampling consisted of randomly removing instances from the majority class (non-MW) until the two classes were balanced. Oversampling consisted of using the Synthetic Minority Over-sampling Technique (SMOTE) algorithm [5]. We also built models without any resampling for comparison purposes.

Table 3. Confusion matrices for best models

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Actual</th>
<th>Classified</th>
<th>Prior</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Global</td>
<td>.65 (hit)</td>
<td>.35 (miss)</td>
<td>.25</td>
</tr>
<tr>
<td>Local</td>
<td>.67 (hit)</td>
<td>.33 (miss)</td>
<td>.26</td>
</tr>
<tr>
<td>Global + Local</td>
<td>.68 (hit)</td>
<td>.32 (miss)</td>
<td>.25</td>
</tr>
</tbody>
</table>

Note: Values are proportionalized by class label
FA = false alarm; CR = correct rejection

Tolerance analysis was performed to address multicollinearity prior to building each model [9]. This consisted of removing features with a tolerance below .2, which indicates highly collinear features (such as number of fixations and number of saccades).

3.5 Model Validation and Evaluation

The models were evaluated using leave-one-student-out cross-validation, which ensures that data from each student is exclusive to either the testing set or training set. Feature selection and resampling were performed on the training set only. Feature selection was performed with data from a random 66% of students in the training data in each fold. Feature rankings were summed over five different random selections. Resampling was also repeated for five iterations in each training fold.

Models were evaluated using the $F_1$ score for the target class (MW), which was compared to the MW $F_1$ score of a chance classifier. For example, if the actual model classified 52% of the instances as MW, the chance classifier would classify a random 52% of the instances as MW. This resulted in a chance precision equal to the actual base rate of MW and a chance recall equal to the predicted MW rate. We believe this chance model to offer a more stringent comparison than a simple minority baseline (assign MW to all instances).

4. RESULTS

4.1 MW Detection Accuracy

The overall best performing model achieved a MW $F_1$ score of .45, compared to a chance MW $F_1$ score of .35, which is consistent with a 29% improvement above chance (Table 2). The model was a decision table classifier that used local features and had a window size of 20 seconds, segment size of 65 seconds, 11 features, and a downsampled training set. The confusion matrix for the model (Table 3) shows that the model makes fewer misses than false alarms.

Table 2. Performance metrics ($F_1$) for best models

<table>
<thead>
<tr>
<th>Feature</th>
<th>$F_1$ MW (Chance)</th>
<th>$F_1$ MW</th>
<th>$F_1$ Non MW</th>
<th>$F_1$ Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global</td>
<td>.35</td>
<td>.39</td>
<td>.57</td>
<td>.53</td>
</tr>
<tr>
<td>Local</td>
<td>.35</td>
<td>.45</td>
<td>.64</td>
<td>.59</td>
</tr>
<tr>
<td>Global+ Local</td>
<td>.36</td>
<td>.39</td>
<td>.54</td>
<td>.50</td>
</tr>
</tbody>
</table>

The best global and global + local models were SVMs with a window size of 15 seconds, a segment size of 65 seconds, and a downsampled training set. The global model contained 5 features, while the global + local model contained 11 features. Both models achieved a lower MW $F_1$ score than the local feature model, due to much higher false alarm rates (see Table 3 and Figure 3).

With respect to the external parameters, no clear trends were observed for window size, segment size, or proportion of features selected, but downsampling and SMOTEnig the training set outperformed no resampling method.

Figure 3. MW $F_1$ score for the best model by feature type and resampling method. G = Global, L = Local, G + L = Global + Local; Down = Downsampling

4.2 Feature Analysis

We compared the mean values of each feature (computed per participant) for MW vs. non-MW instances with a two-tailed paired-samples t-test. We focused on the 16 global and 21 local features that were included in the best local and global models. Table 4 shows the effect size (Cohen’s $d$ – with positive values of $d$ denoting higher values for MW compared to non-MW instances) for the significantly different ($p < .05$) features. We did not perform adjustments for multiple comparisons as the present analysis is exploratory in nature. Further, the number of significant findings (18%) is far greater than what we could achieve if we were capitalizing on chance alone.

We note that students were less likely to focus on the AOIs when they were MW. This is evidenced by a fewer number of frames where the smooth pursuit points intersected with the red balloon AOI or the most visually salient AOI. Further, there were fewer saccades onto and off of the most visually salient region during MW. Third, smooth pursuits had a longer range, but less variability in velocity during MW. Finally, there were fewer saccades during MW, which is consistent with previous findings of eye movements during MW while reading [2, 28]. Taken together, these results reflect a decoupling between salient regions...
on the screen and eye movements, essentially signaling a breakdown in attentional synchronicity during MW.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Cohen’s d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smooth Pursuit with Balloon AOI (frames)</td>
<td>-.37</td>
</tr>
<tr>
<td>Smooth Pursuit within 2° Saliency AOI (frames)</td>
<td>-.38</td>
</tr>
<tr>
<td>Number of Saccades away from Saliency AOI</td>
<td>-.39</td>
</tr>
<tr>
<td>Number of Saccades nearly onto Saliency AOI</td>
<td>-.35</td>
</tr>
<tr>
<td>Smooth Pursuit Duration Range (ms)</td>
<td>.30</td>
</tr>
<tr>
<td>Smooth Pursuit Velocity SD (%)</td>
<td>-.28</td>
</tr>
<tr>
<td>Number of Saccades</td>
<td>-.31</td>
</tr>
</tbody>
</table>

Note: SD = Standard Deviation; All tests were significant at p < .05 df = 53 for local features and df = 50 for global features.

5. DISCUSSION

There is a growing interest in assuaging the negative effects of MW during learning [6, 8]. Reliable MW detection is likely required to realize this goal. Although efforts in MW detection have had some success in the context of reading, MW detection in more media-rich contexts has been unexplored. As a step in this direction, this paper presents a student-independent detector of MW during narrative film comprehension, a context which is both timely and relevant given the increasing use of film and video lectures as educational resources.

5.1 Key Findings and Contributions

Our primary contribution is the computation of novel local gaze features that are based on the dynamic visual content of the film. Using these features, we were able to detect MW with a F1 of .45 reflecting, a 29% improvement over chance. Furthermore, models built with local features outperformed models built with global features, or a combination of both global and local features. This suggests that taking the dynamic visual content into account (local features) can be more effective than merely tracking overall gaze patterns (global features), which has been the common method for MW detection during reading.

The local features likely performed better in the present context (narrative film viewing) compared to reading, because the unfolding visual stream provides cues as to where attention should be directed. Reading, in contrast, does not provide such explicit cues, so there is likely more variability in gaze patterns. This would explain why the global gaze features outperformed the local features during reading.

We also found that local features outperformed a combined local + global model, but we adopted a rather simplistic feature-level fusion strategy. It is an open question as to whether performance of the combined model could be boosted with more advanced fusion strategies.

Our results also provide insight into eye movements related to MW during film viewing. The key finding was that eye movements during MW were decoupled from the visually salient and important (balloon AOI) components of the visual stream, suggesting a breakdown in attentional control.

5.2 Applications

MW impedes comprehension by diverting a student’s attention from the task at hand toward task-unrelated thoughts. Educational activities that involve comprehension from dynamic visual scenes, such as video clips or short instructional lectures, could benefit from pairing a MW detector with interventions that direct attention toward the learning task.

Beyond educational interfaces, detectors built from dynamic visual scenes have applications in entertainment and safety contexts. For example, they could be used to determine when viewers are more likely to MW while viewing entertainment films. The scenes could then be improved to increase viewer engagement.

Attentional focus is especially important for safety-critical tasks that require vigilance, such as air traffic control. MW detectors built for dynamic visual scenes might be more suitable for these types of tasks. However, empirical evidence is needed to determine the extent to which models built from narrative film viewing would generalize to these other contexts.

5.3 Limitations and Future Work

There were also some limitations with this study. The first limitation is the detection accuracy, which is moderate at best. It would be fruitful to explore improvements to the detector. Some possibilities include considering additional features based on other aspects of the visual content, such as faces or attempting more sophisticated modeling approaches that capture the unfolding temporal dynamics in eye gaze.

The segmentation method used in the study reflects yet another limitation as it rather arbitrarily segments the visual stream based on temporal windows. It would be worthwhile to explore content-based segmentation, such as scene transitions and event boundaries. This would also ensure consistent segments across students in lieu of the current method, which segments the film at different locations depending on the MW reports.

It is also unclear if the detector would generalize beyond the current film. “The Red Balloon” is a commercially produced film that employs cinematic devices to draw attention to the viewer [3]. In contrast, many instructional videos consist of an instructor lecturing to students [13] or lecturing over power point, which reflect rather different visual content.

Another limitation is the cost of eye tracking technology. The eye tracker used for this study was a cost-prohibitive Tobii TX300 that will not scale out of the laboratory. Fortunately, cost-effective eye tracking alternatives are becoming available, such as the Eye Tribe and Tobii EyeX, so replication with these trackers is warranted.

Finally, other limitations include a limited student sample (i.e. undergraduates from a private Midwestern college) and a laboratory setup. It is possible that the detector would not generalize to a more diverse student population or in more ecological environments. Retraining our model with data from more diverse populations and environments would be a suitable next step to increase its ecological validity.

5.4 Conclusion

We built the first student-independent gaze-based MW detector in the context of film viewing. The detector could be used to trigger interventions aimed at counteracting the negative effects of MW for an array of tasks involving dynamic visual scenes (e.g., watching instructional films, historic documentaries, or video lectures). Taken together, this work takes us closer to the goal of developing next-generation intelligent educational interfaces that “attend to attention” [6].
6. ACKNOWLEDGMENTS

This research was supported by the National Science Foundation (NSF) (DRL 1235958 and IIS 1523091). Any opinions, findings and conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF.

7. REFERENCES


Riding an emotional roller-coaster: A multimodal study of young child’s math problem solving activities

Lujie Chen
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA USA
lujiec@andrew.cmu.edu

Xin Li
Shandong Yingcai University
No. 2 Yingcai Road
Shandong, China
lixin@ycxy.com

Zhuyun Xia
Shandong Yingcai University
No. 2 Yingcai Road
Shandong, China
xiazhujun@ycxy.com

Zhanmei Song
Shandong Yingcai University
No. 2 Yingcai Road
Shandong, China
songzhanmei@ycxy.com

Louis-Philippe Morency
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA USA
morency@cs.cmu.edu

Artur Dubrawski
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA USA
awd@cs.cmu.edu

ABSTRACT
Solving challenging math problems often invites a child to ride an “emotional roller-coaster” and experience a complex mixture of emotions including confusion, frustration, joy, and surprise. Early exposure to this type of “hard fun” may stimulate child’s interest and curiosity of mathematics and nurture life long skills such as resilience and perseverance. However, without optimal support, it may also turn off child prematurely due to unresolved frustration. An ideal teacher is able to pick up child’s subtle emotional signals in real time and respond optimally to offer cognitive and emotional support. In order to design an intelligent tutor specifically designed for this purpose, it is necessary to understand at fine-grained level the child’s emotion experience and its interplay with the inter-personal communication dynamics between child and his/her teacher. In this study, we made such an attempt by analyzing a series of video recordings of problem solving sessions by a young student and his mom, the ideal teacher. We demonstrate a multimodal analysis framework to characterize several aspects of the child-mom interaction patterns within the emotional context at a granular level. We then build machine learning models to predict teacher’s response using extracted multimodal features. In addition, we validate the performance of automatic detector of affect, intent-to-connect behavior, and voice activity, using annotated data, which provides evidence of the potential utility of the presented tools in scaling up analysis of this type to large number of subjects and in implementing tools to guide teachers towards optimal interactions in real time.

Keywords: math problem solving, affect, interaction dynamics, multimodal learning analytics

1. INTRODUCTION
A popular perception of math education in the US schools is often associated with the lack of inspiration and excitement. One of the possible reasons for that is a common perception of math learning as shallow learning activities such as memorizing multiplication tables and procedure learning activities such as long division [10]. This is especially true with elementary level education where learning facts and procedures accounts for most of the curriculum. In contrast, math problem solving activities can take a form of complex learning [10] that often requires the student to take an adventurous emotional and cognitive “roller-coaster” ride when navigating the uncharted land of possible solutions.

Involvement in this type of activities from young age may play a major role in stimulating student’s interest in math and more generally in STEM topics. It may also help building self-confidence and perseverance. However, if not done right, it may disengage student due to unresolved frustration and result in an even more negative view of the subject. It is thus important to know what is the right mixture of emotional and cognitive support to be provided in the process, as well as the right amount and the optimum timing of such support. This role of support is consistent with the vision of a Learning Companion [12] which is a computer system that facilitates learning on the side, is watchful for the trajectory and provides appropriate level of support.

In this study, we explore that question by analyzing the fine-grained multimodal behavior cues that could be automatically extracted from video recordings of one-to-one math problem solving sessions in a naturalist environment. Specifically, we explore data driven methods to characterize the temporal dynamics of the child’s emotion states as well as patterns of the interaction between the child and the teacher when problem solving processes unfold.

2. RELATED WORK
A substantial amount of prior work on the automatic detection of student’s affective states exists primarily in the context of intelligent tutor systems. [2] introduces a “sensor free” detector to infer engagement from the logs of students’ interaction with computerized reading tutor using a method called engagement tracing. [15] uses facial expression analysis to infer engagement during interactive cognitive skill training sessions. Using the same sensing modality, [13] studies an array of affective states such as boredom, confusion, delight, flow, frustration and surprise, based on Facial Action Units. [5] leverages multimodal inputs including conversational cues with computer tutors and gross body language as well as facial features to detect distinct affective states.

While the work mentioned above focuses on static modeling of affects, another thread studies dynamics of affective states. [5] characterizes transitions of affective states between confusion, engagement/flow, boredom and frustration during complex learning activities when using computer tutors. [11] uses a hierarchical dynamic Bayesian network to model temporal dynamics of behavior trends such as flow, stuck and off-task, as well as related emotion states such as stress, confusion, boredom and frustration.

Within literature on student and human teacher interaction, [14] applied theory of dynamic systems to model real time teacher-student interactions using videotaped classroom sessions. Quality of interaction was rated and analyzed in terms of content, structure and complementary. [8] uses turn level audio features and contextual information to predict students’ high level affect states using a human-human tutoring dialogue corpus.

There are several aspects in which this study differs from relevant prior work: (1) Instead of using computer tutor, we are interested in an “unplugged” scenario where the child is interacting with a real human teacher. This setup allows us to observe the genuine inter-personal communication dynamics which is not available when interacting with a computer tutor. Specifically, help seeking behaviors, a well studied phenomenon with computer tutors, are generalized into Intent-To-Connect (ITC) behaviors manifested by either subtle cues such as eye contacts or head pose changes, or explicit verbal help requests. ITC behaviors carry a richer meaning that exceeds the conventional cognitive support oriented “help seeking”. Instead, ITC behaviors can also be used to signal emotional connection for other purposes such as “comfort seeking” or “joy sharing”; (2) The subject in this study is a child at young age. Since children at this age often are not exposed to the social pressure to hide negative emotions such as frustration, this allows observing their emotions with high fidelity, though it also presents unique detection challenges since the frequent baseline body movement are more frequently observed in young children; (3) The problem solving tasks in this study call for the child to take an active role in open exploration, with support from adult only when needed, whereas other studies typically consider a specific task such as cognitive skill training [15]. Consequently, we expect to observe non-baseline affect states at higher level of frequency and intensity; (4) With a few exceptions, most of the existing work relies on signals from a single modality, while this study attempts to integrate multimodal signals available from audio and video data.

3. DATASET AND USER STUDY

We collected video recordings of one-to-one problem solving sessions between a 9-year-old boy (a third grader) and his mom (the first author of this paper) as his teacher. We chose this setup because this mom and son has worked together on math problem solving for a few years. As result, the mom is used to picking up and reacting optimally to child’s behaviors. This is the closest to the desirable model of the “ideal teacher” as we described earlier.

In each of multiple sessions, the child was asked to solve one challenging math problem. We selected the problems from Math Kangaroo1, an annual international math competition for students in K-12. Using interesting but challenging problems, the goal of this competition is to stimulate students’ interest in math problem solving. There are 24 problems in each competition, divided into three sections with gradual increase of difficulty. The problems for this study were selected from the most difficult set of levels 3 and 4 competition geared towards students in third and fourth grades. Those problems assume basic arithmetic skills and background knowledge at the child’s grade level. Figure 1 shows an example of a problem used in the study. In all of the sessions, mom tried to optimize the experience of the child by balancing the goal of reducing frustration and providing sufficiently stimulating challenge.

The videos were captured in a home environment using a Logitech 1080P webcam with an integrated microphone. The positions of mom and child make it possible to capture child’s non-verbal behavior cues such as head pose and gaze changes when he intends to connect with mom. Both audios and videos were captured for child, whereas only voice was recorded for mom. We recorded a total of 21 sessions, accumulating 141 minutes of raw video with mean length of 6.4 minutes per session, with longest session lasting 14.6 minutes and the shortest only about 2 minutes. In most of the recordings, the child ended with a joyful mood and a sense of accomplishment.

All recordings were manually annotated in ELAN 2[3] for voice activity at utterance level of child and mom. We also annotated child’s non-verbal ITC behaviors using cues such as head turn and eye contact as well as verbal cues. Annotation included timestamps of start and end of events. Frame-
by-frame emotion states were extracted using FACET Software Development Kit. Head pose and gaze features were extracted using OpenFace framework toolkit [1]. In addition, acoustic features were extracted using COVAREP toolkit (version 1.3.2) [4] every 10ms.

4. QUALITATIVE ANALYSIS

4.1 Problem solving stages and affective states

In his famous book “How to solve it” [9], the mathematician George Polya proposed four stages of problem solving, a framework widely used in today’s math problem solving instructions. In this study we adapt it into a three-stage framework without the last reflection stage, including “problem understanding”, “planning” and “execution”. Table 1 lists the most likely affective states as well as plausible behavior cues at each problem solving stage based on qualitative analysis of video recordings. Those cues were used to guide the annotation of events as well as informed the feature design for the automated analysis.

<table>
<thead>
<tr>
<th>Affects</th>
<th>Problem Understanding</th>
<th>Planning</th>
<th>Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confusion</td>
<td>F+Ver</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frustration</td>
<td>F+HP+Voc</td>
<td>F+HP</td>
<td></td>
</tr>
<tr>
<td>Joy</td>
<td>F+Ver</td>
<td>Voc+HP</td>
<td></td>
</tr>
<tr>
<td>Engaged</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disengaged</td>
<td>HP</td>
<td>HP</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: The affective state and problem solving stages and their behavior cues F(facial), HP(head pose), Voc(vocal), Ver(verbal)

There are several “landmark” behavior cues that could be used to identify problem solving stages and transitions. During problem understanding stage, the child reads the problem and asks clarification questions when necessary. The child often ends this stage by saying “okay”. Afterwards, the child might be stuck at the planning stage with no idea as for how to proceed, or go on smoothly with a brief planning stage, or in rare cases dive right into the implementation stage. During the implementation stage, the child is often engaged, with his head down, writing on paper, either silently or with fast paced talking suggesting a “flow” experience. After one attempt, he may succeed at solving the problem, or he could find that his answer is obviously wrong. In those cases, he needs to re-enter into the planning stage to find alternative solution, or rework the original plan. The process ends when the correct answer is confirmed in which case the child often exhibits positive emotions such as excitement and joy.

4.2 Interpersonal communication dynamics

The problem solving sessions can be highly interactive between mom and child: the child actively verbalizes his problem solving process and frequently connects with mom through verbal and non-verbal cues which we call “intent-to-connect” behaviors, or, ITC. Verbal ITC cues refer to explicit request for help or questions, while non-verbal ITCs are subtle cues of head pose and/or gaze change.

ITC may carry multiple different meanings, which calls for differentiated responses to achieve best learning outcomes. According to her interpretation, mom’s response to ITC may serve a purely cognitive support purpose such as providing scaffolding, or, as in most cases, providing emotional support in the form of “back channel” signals such as “yes”, “good”, “good thinking”. Given the many subtle variation of ITCs that can be considered in modeling response, it is desirable to take into account contextual information such as problem solving stages and emotion states in order to infer the true intent of an ITC.

Figure 2 provides an overview of the events of an example session that illustrates the interplay between interpersonal communication dynamics, including voice activity events (mom’s talk and child’s talk) and child’s ITC behaviors, within the context of problem solving stages transitions and emotion states. As shown in the plot, the session started with the problem understanding stage (1) that is characterized by child’s monologue while reading the problem followed by a brief period of pause and thinking. At the same time, confusion and frustration began to kick in (A), after which mom started to intervene by explaining the problem (2), then child entered planning and execution stage (3) that lasts about 3 minutes. Then, at 1 minute into this process, child said “I didn’t get it” with head turn, and mom offered help by asking “Do you need help?”. However, the child did not take the offer and kept working on his own. Towards the end of this phase, the child exhibited positive emotion of joy. Then mom discovered that child is on the wrong path, so she intervened (4) and the two worked together to correct the error during which time the child showed brief moments of frustration and confusion (C). Afterwards, the session moved into the problem solved stage (5), the child revealed a spike of surprise and moderate joy (D).

5. QUANTITATIVE ANALYSIS

In this section, we present an analytic framework developed to characterize and understand the interplay between dynamics of emotional states as well as interpersonal communication. We first present a method to quantify the relationship between ITC and mom and child’s talk. We then present results from analysis of videos using emotion and interaction features. We end this section with predictive modeling of mom’s response using multimodal features.

5.1 Interpersonal communication dynamics

5.1.1 Event intensity metric

We use event intensity metric to characterize temporal patterns of intensity of a specific type of event (e.g. child’s talk). This metric takes into account both the frequency and duration of an event. To compute the metric, we first convert the annotated duration of the events into discrete sequences sampled uniformly at interval of every 20ms. Binary flag of 1 is assigned to intervals of the event’s occurrence and 0 otherwise. A moving sum is then computed from a window centered at the time of interest. The resulting time series of the moving sum of thusly assigned binary flags characterizes
temporal intensity distribution of the events. The width of the window determines temporal resolution and smoothness of the temporal patterns.

5.1.2 Floor sharing metrics
We characterize temporal patterns of floor sharing between mom and child using normalized metrics of event intensity of mom’s talk and child’s talk as described above. The formula for mom’s sharing of conversation at time stamp $t$ is given as:

$$\text{Mom Talk Share}(t) = \frac{\text{Mom Talk}(t)}{\text{Child Talk}(t) + \text{Mom Talk}(t)}$$

This metric is useful to identify periods of time when mom’s intervention dominates or vice versa. Figure 3 shows temporal distribution of floor sharing patterns for each video sorted by video length. It seems apparent that in short videos (presumably representing easy problems), mom did not talk much. However, longer videos often involve larger proportion of mom’s talk. It is also interesting to observe that mom’s talk often occurs in batches, presumably at the time when child gets stuck so that elaborate explanation is necessary.

5.1.3 Synchronization of voice activity and ITC
In this section, we describe a method to quantify synchronization between voice activity (mom’s talk and child’s talk) and ITC. Figure 4 shows two examples with different synchronization patterns. In the left plot, ITC seems to be more synchronized with child’s talk, while in the right plot it is more synchronized with mom’s talk which suggests child’s attention or engagement. We summarize synchronization as the pairwise correlation among these time series. The result is displayed in the scatter plot in Figure 5 in which each video is plotted as a point labeled with its index. As shown, ITC seems to be more correlated with mom’s talk than child’s talk as seen from the cluster of points in the upper left quadrant of the plot in Figure 5, with a few exceptions (videos 12, 14 and 32) in which ITC seems to be drifted away from mom’s talk and correlate more with child’s talk. Incidentally, mom intervened significantly in those videos, which suggests child’s disengagement may be induced by mom’s higher intensity of teaching.

5.2 Video analysis
In this section, we report the results from video analysis by exploring the pairwise statistical correlations among variables related to interaction dynamics (i.e. voice activity and ITC behaviors) and affective states, as well as the outcome measure, i.e. time taken to solve a problem. For each video, we computed the following variables:

1. Interaction dynamics variables

   - Mom/Child talk ratio (mom-child): The ratio of the accumulative duration of mom’s talk versus child’s talk.
   - ITC rate: The count of ITC, normalized by video length.
Figure 3: Temporal patterns of floor sharing for each video (dark color: mom, light color: child)

Figure 4: Two example time series plot of events intensity, ITC synchronized more with child’s talk (left) or with mom’s talk (right)

Figure 5: Summary of synchronization: ITC vs child talk (x axis) and ITC vs mom talk (y axis)
• Mom’s back channel response rate (mom-BC): Back channel response is defined as a response that lasts less than 2 seconds. This variable represents the count of such response normalized by video length.

2. Affective state variables

• These are counts of video frames with FACET score greater than 1, normalized by total number of frames during the period of interest for each of the four affect channels including joy, surprise, frustrations and confusion.

In order to further explore the importance of features at the beginning as well as those at the end of a session, we also compute statistical features from two sub-periods of interest: first 30 and last 30 seconds of each video.

We then compute pairwise Pearson correlation among the variables, including outcome. Due to the small number of videos, for each pair of correlations, we performed 1000 iterations of a randomization test [7] under null hypothesis of zero correlation to obtain non-parametric p-values. A sparse graph (Figure 6) is created to summarize the significant correlations among the variables with a p-value cutoff at 5% significance level.

There are several interesting insights that could be derived from this graph. Firstly, there is a significant positive correlation between initial frustration or confusion and the time taken to solve a problem. Since the beginning period is likely to be devoted to problem understanding, this suggests difficulty in understanding of the problem is the first obstacle child may face. His struggle in this period is likely to extend over the entire problem solving process. Secondly, there is a positive correlation between mom/child talk ratio and the video length. This suggests that mom intervenes more in case of hard problems which take longer to solve. Thirdly, child’s ITC rate is positively correlated with mom’s back channel rates which suggests a level of interaction synchrony between the two. Lastly, there is negative correlation between the overall frustration and joy at the ending period, in other words, more frustrating experience is associated with less joy toward the end, and vice versa.

5.3 Predictive modeling of response

In this section, we report the results from machine learning models used to predict the binary label if there is mom’s response within 5 seconds for occurrence of an ITC. The following list explains the features used for the predictive model:

1. Voice activity features:
   • ITC co-occurrence: The count of other ITC within time windows of 2, 5 and 10 seconds respectively for each ITC;
   • Overlap statistics: The number of child talk, mom talk and child or mom talk events that are overlapping a given instance of ITC;

   Table 2: Performance of the predictive models of mom’s response to child’s ITC (leave one video out)

<table>
<thead>
<tr>
<th>Model</th>
<th>AUC mean</th>
<th>Lower bound of CI</th>
<th>Upper bound of CI</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>0.594</td>
<td>0.557</td>
<td>0.630</td>
</tr>
<tr>
<td>NB</td>
<td>0.617</td>
<td>0.581</td>
<td>0.652</td>
</tr>
<tr>
<td>SVM</td>
<td>0.519</td>
<td>0.506</td>
<td>0.531</td>
</tr>
</tbody>
</table>

2. Head pose features: Min, max, mean, median of detection success, confidence, tilt, turn, up-down, within 5 seconds surrounding a given ITC;

3. Features from affect detector: Min, max, mean, median of FACET score for each of the emotion categories (joy, surprise, confusion, frustration and baseline) within the 5 seconds surrounding a given ITC. Negative scores are replaced with 0.

We performed a leave-one-video-out cross-validation experiment to evaluate three different classifiers (logistic regression[LR], naive bayes[NB] and support vector machine[SVM]). The Area Under Curve(AUC) score for each classifier is shown in Table 2 with mean values and 95% confidence intervals. Though the overall performance has much room for improvement, all of the three models perform significantly better than random, which suggests there are indeed predictive signals in the features. A better model might need to incorporate features related to the problem solving state, which may be learned using state space method such as Hidden Markov Models or Conditional Random Fields.

6. VALIDATION OF AUTOMATIC RECOGNITION

6.1 ITC and voice activity recognition

In this section we summarize the results from following recognition tasks:

1. ITC recognition using Openface head pose features. For each video, a random sample of 500 positive frames with ITC and 500 negative frames without ITC were selected, and a model was trained using frame-by-frame head pose features (confidence, Tx, Ty, Tz, Rx, Ry, Rz, up-down, turn and tilt) as inputs;

2. Voice activity recognition using features from COVARAP. One classifier built to discriminate between speaker and non-speaker segments; another classifier to discriminate mom’s talk and child’s talk. For each task, we random select 500 samples from each class from each video.

In those recognition tasks, we experimented with different types of classifiers including logistic regression, support vector machine, decision tree and naive Bayes, and found logistic regression to show overall superior performance as reported in Table 3. We performed leave-one-video-out cross validation and reported mean AUC scores. We also reported per video performance where we build a dedicated classifier for each video and summarized 10-fold cross-validation AUC score across all videos. As expected, leave-one-video performance is worse than the per video performance for both ITC
Figure 6: Graph of pairwise correlation of variables. Variable named in the form of “xxx” (e.g. joy) are computed from full length of video; “xxx_b” (e.g. mom_child_b) are computed from first 30 seconds of each video, “xxx_e” (e.g. surprise_e) are computed from the last 30 seconds. Black edges depict positive correlations while red edges represent negative correlations. The width of the edge corresponds to the magnitude of the absolute value of the correlation. The colors of the nodes denote types of variable: Green-Joy, Red-Frustration, Golden-Surprise, Light Yellow-Confusion, Gray-Interpersonal dynamics features, Blue-Outcome

Table 3: AUC scores of models built for ITC and voice activity recognition task

<table>
<thead>
<tr>
<th></th>
<th>ITC recognition</th>
<th>Speaker vs. non-speaker</th>
<th>Mom and child talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leave one video out CV</td>
<td>0.90</td>
<td>0.81</td>
<td>0.74</td>
</tr>
<tr>
<td>Dedicated classifier 10-fold CV</td>
<td>0.92</td>
<td>0.81</td>
<td>0.81</td>
</tr>
</tbody>
</table>

Table 4: Validation scores of FACET’s affect detection (Cohen’s Kappa)

<table>
<thead>
<tr>
<th>Affect</th>
<th>annotator1 vs FACET</th>
<th>annotator2 vs FACET</th>
<th>annotator1 vs annotator2</th>
</tr>
</thead>
<tbody>
<tr>
<td>joy</td>
<td>0.70</td>
<td>0.57</td>
<td>0.73</td>
</tr>
<tr>
<td>surprise</td>
<td>0.48</td>
<td>0.43</td>
<td>0.71</td>
</tr>
<tr>
<td>confusion</td>
<td>0.30</td>
<td>0.51</td>
<td>0.41</td>
</tr>
<tr>
<td>frustration</td>
<td>0.11</td>
<td>0.36</td>
<td>0.44</td>
</tr>
<tr>
<td>baseline</td>
<td>0.58</td>
<td>0.42</td>
<td>0.44</td>
</tr>
<tr>
<td>overall</td>
<td>0.35</td>
<td>0.46</td>
<td>0.41</td>
</tr>
</tbody>
</table>

6.2 Affect detection
In this section, we report validation results for affect labels produced by FACET. We randomly selected 30 top-scored frames (at least 10 seconds apart) from each of the affect class (joy, surprise, frustration, confusion and baseline), and requested labels from two independent annotators who were blinded from FACET labels. Table 4 shows Cohen’s Kappa for each affect label (when treated as a binary labeling task) as well as the overall score. As shown, the inter-rater agreement is relatively high for both joy and surprise, though the annotator’s agreement with FACET is higher for joy than surprise. Confusion and frustration are two of the most challenging affects to detect as compared to joy and surprise, possibly due to the fact that confusion and frustration are easily mistaken for each other, as evidenced by the low inter-rater agreement score. This suboptimal performance may also be attributable to the fact that FACET is trained on faces from general population rather than specifically on young children. A detection algorithm that would incorporate transfer learning and age based customization will possibly improve the performance.

7. CONCLUSION AND FUTURE WORK
In this study, we analyzed data from the 21 video recordings of a nine year old boy while he was working through challenging math problems that demand high order cognitive skills to understand, plan, execute and solve the problems on his own, with only limited and mostly passive support from his mom.
We have shown qualitatively that there are clusters of non-baseline emotions rolling throughout the problem solving process, with the strongest representation from emotion class of joy, surprise, confusion and frustration. This observation confirmed our hypothesis that this type of active exploration indeed facilitates a unique experience of riding an “emotional roller coaster”.

We also explored various analytical approaches to characterize the interpersonal dynamics between mom and child as well as the interplay with ITC behaviors. Our video analysis reveals some interesting associations between voice activity, ITC and emotional context.

Lastly, we built a classification model to predict whether there is mom’s response within 5 seconds of a given ITC. The recognition task results show promise for automatic annotation of ITC and voice activity in order to scale up the presented analysis. Those findings collectively provide initial evidence for the feasibility of building affect sensitive computer tutor by mining multimodal signals as demonstrated in this study.

The key contributions of this paper include the new framework for fine-grained analysis of affect dynamics during student’s interaction with a human teacher, the use of multimodal signals in truly dynamic settings, and demonstration of the utility of the proposed approach to automatically detect behaviors and predict emotions.

We consider multiple thrusts of future work. With the current data set, we envision the following tasks worth consideration: (1) Learn latent dynamic model for problem solving state recognition so that it can be used to improve predictive model of ITC; (2) Explore the possibility of automatic transcription with Automatic Voice Recognition system, and explore sentiment analysis of mom’s response; (3) Explore the utility of prosody features of speech signals to complement the current visual cues based affect detection. Another research direction involves extending this study to more subjects so that inter-subject variation can be observed and modeled. In addition, we would also like to explore the possibility of transferring models learned from one child to another. It is also of interest to explore the correlation between metrics gathered in this study with psychological instruments such as grit scales [6]. Last but not least, we envision our current work to be a foundation for a future tool to guide teachers towards optimal interactions with their students in real time.

8. ACKNOWLEDGMENTS
We would like to thank Liangke Gui for help extracting features from FACET and COVARAP and Tadas Baltrusaitis for helping with Openface features. This work has been partially supported by NSF (1320347).

9. REFERENCES
Joint Discovery of Skill Prerequisite Graphs and Student Models

Yetian Chen⋆†, José P. González-Brenes†, Jin Tian

ABSTRACT
Skill prerequisite information is useful for tutoring systems that assess student knowledge or that provide remediation. These systems often encode prerequisites as graphs designed by subject matter experts in a costly and time-consuming process. In this paper, we introduce Combined student Modeling and prerequisite Discovery (COMMAND), a novel algorithm for jointly inferring a prerequisite graph and a student model from data. Learning a COMMAND model requires student performance data and a mapping of items to skills (Q-matrix). COMMAND learns the skill prerequisite relations as a Bayesian network (an encoding of the probabilistic dependence among the skills) via a two-stage learning process. In the first stage, it uses an algorithm called Structural Expectation Maximization to select a class of equivalent Bayesian networks; in the second stage, it uses curriculum information to select a single Bayesian network. Our experiments on simulations and real student data suggest that COMMAND is better than prior methods in the literature.

Keywords
Prerequisite discovery, Bayesian network, student modeling

1. INTRODUCTION
Course curricula are usually organized in a meaningful sequence that evolves from relatively simple lessons to more complex ones. Among these lessons, some are required to be mastered by the student before the subsequent ones can be learned. For instance, students have to know how to do addition before they learn to do multiplication. We refer to prerequisite structure as the relationships among skills that place strict constraints on the order in which skills can be acquired.

Prerequisite structures are crucial for designing intelligent tutoring systems that assess student knowledge or that offer remediation interventions to students. Building such systems require prerequisite information that is often hand-engineered by subject matter experts in a costly and time-consuming process. Moreover, the prerequisite structures specified by the experts are seldom tested and might be unreliable in the sense that experts may have “blind spots”.

Recent interest in computer assisted education promises large amounts of data from students solving items—questions, problems, parts of questions. Performance data (what items a learner answers correctly) can be used to create student models. These models represent an estimate of skill proficiency at a given point in time. For example, a student model can represent that Alice has already mastered integer addition, but Bob has not. Student models are often used to personalize instruction in tutoring systems or to predict future student performance. In this paper, we introduce Combined student Modeling and prerequisite Discovery (COMMAND), a novel algorithm for simultaneously discovering prerequisite structure of skills and a student model from student performance data.

2. RELATION TO PRIOR WORK
Prior work has investigated how to discover prerequisites among items without considering their mapping into skills. Item-to-skill mappings (also called Q-matrices) are desirable because they allow more interpretable diagnostic information. Because of this, follow-up work has studied whether a pair of skills have a prerequisite relationship or not. For this, we can measure if a model that assumes a dependency between the two skills explains the data better than a model that assumes independence. This comparison can be done with data likelihood or association rule mining. Although promising, prior methods have limitations that we address:

1. We estimate the global prerequisite structure, not just the pairwise relationships. For example, suppose we want to discover the prerequisites of three skills for English learning (S1: syntax, S2: cohesion and S3: lexical rules). If we use prior methods, we discover that the three skills are related among each other. However, pairwise methods are unable to tell if the relationships are due to indirect (e.g., S1 → S2 → S3), or direct (e.g., S1 → S2 → S3) effects.

2. It is unclear how to use the output of these prerequisite structures for student modeling. For example, it is not obvious how to best use them to make predictions of future student performance.

3. Prior work does not provide quantitative evaluation using real student data. Overall, learner data has been used to provide examples, but without any methodology that can help compare what algorithm works better.

A statistical formalism called Bayesian network has been useful to model prerequisite structures. Bayesian networks allow modeling the full structure of skills (beyond pairwise relationships)
and can encode conditional independence between the skills. Unfortunately, prior work with Bayesian networks requires a domain expert to design the prerequisite structures [10], and automatic techniques have not been demonstrated with real student data [14]. We now describe the COMMAND algorithm that discovers a Bayesian network that encodes the prerequisite structure of skills.

### 3. THE COMMAND ALGORITHM

COMMAND learns the prerequisite structure of the skills from data with a statistical model called Bayesian network [13, 15]. Bayesian networks are one type of probabilistic graphical models because they can be represented visually and algebraically as a collection of nodes and edges. A tutorial description of Bayesian networks in education can be found elsewhere [12], but for now we say that they are often described with two components: the nodes represent the random variables, which we describe using conditional probability tables (CPTs), and the set of edges that form a directed acyclic graph (DAG) represent the conditional dependencies between the variables. Bayesian networks are a flexible tool that can be used to model an entire curriculum.

Figure 1 illustrates an example of a prerequisite structure modeled with a Bayesian network. Here, we relate four test items with the skills of addition and multiplication. Addition is a prerequisite of multiplication thus there is an arrow from addition to multiplication. Modeling prerequisites as edges in a Bayesian network allows us to frame the discovery of the prerequisite relationships as the well-studied machine learning problem of learning a Bayesian network from data with the presence of unobserved latent variables. We represent the prerequisite structure using Bayesian networks that use latent binary variables to represent the student knowledge of a skill (i.e., mastery or not mastery), and observed binary variables that represent the student performance answering items (i.e., correct or incorrect).

Algorithm 1 describes the COMMAND pipeline. The input to COMMAND is a matrix $D$ with $n \times p$ dimensions, representing $n$ students, answering $p$ items. Each entry in $D$ encodes the performance of a student (see Table 1 for an example). Additionally, we require a $Q$-matrix to represent the item-to-skill mapping. $Q$-matrices are often designed by subject matter experts but automatic methods to discover them exist [8].

![Figure 1: A hypothetical Bayesian network. Solid edges are given by item to skill mapping, dashed edges between skill variables are to be discovered from data. The conditional probability tables are to be learned.](image)

**Table 1: Example student performance matrix to use with COMMAND.** The performance of a student is encoded with 1 if the student answered correctly the item, and 0 otherwise.

<table>
<thead>
<tr>
<th>User</th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
<th>Item $p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Algorithm 1** The COMMAND algorithm

Require: A matrix $D$ of student performance on a set of test items, skill-to-item mapping $Q$ (containing a set of skills $S$).

1. $G_0 \leftarrow \text{Initialize}(S, Q)$
2. $i \leftarrow 0$
3. do
4. $E$-step:
5. $\Theta^* \leftarrow \text{ParametricEM}(G_i, D)$
6. $D^* \leftarrow \text{Inference}(G_i, \Theta^*, D)$
7. $M$-step:
8. $(G_{i+1}, \Theta_{i+1}) \leftarrow \text{BNLearning}(G_i, D^*)$
9. $i \leftarrow i + 1$
10. while stop criterion is not met do
11. $RE \leftarrow \text{FindReversibleEdges}(G_i)$
12. $EC \leftarrow \text{EnumEquivalentDAGs}(G_i)$
13. $DE \leftarrow \{\}$
14. for every reversible edge $S_i \rightarrow S_j$ in $RE$ do
15. $\text{ratio} \leftarrow \frac{P(S_i = 0, S_j = 0)}{P(S_i = 0, S_j = 1)}$
16. if $\text{ratio} \geq 1$ then
17. $\text{ratio}^* = \text{ratio}$
18. $DE \leftarrow DE \cup S_i \rightarrow S_j$
19. else
20. $\text{ratio}^* = \frac{1}{\text{ratio}}$
21. $DE \leftarrow DE \cup S_i \leftarrow S_j$
22. end if
23. end for
24. $\text{sort}(DE)$ by $\text{ratio}^*$ in descending order
25. while $DE$ is not empty do
26. $e \leftarrow \text{dequeue}(DE)$
27. if $\exists G \in EC, e \in G$ then
28. $\forall G \in EC, \text{remove } G$ from $EC$ if $e \notin G$
29. end if
30. end while
31. return $EC$

COMMAND relies on a popular machine learning algorithm called Structural Expectation Maximization (Structural EM), which to the extent of our knowledge has not been used in educational applications before. Structural EM extends the Expectation Maximization (EM) algorithm to allow efficient structure learning of Bayesian networks when there are latent variables or missing values in the data. A secondary contribution of our work is introducing Structural EM for learning Bayesian network structures from educational data. We now describe the steps of COMMAND in detail.

#### 3.1 Initial Bayesian Network

COMMAND first creates an initial Bayesian network using the $Q$-matrix by creating an arc to each item from each of its required

$$P(S_i = a|S_j = b)$$ can be computed using any Bayesian network inference algorithm such as Junction tree algorithm [11].
skills. Because there are no edges between the skills, this initial network does not encode any prerequisite information. COMMAND uses Structural EM to learn arcs (prerequisites) between the skill variables.

3.2 Structural EM

A common solution to learning a Bayesian network from data is the score-and-search approach [5, 9]. This approach uses a scoring function (like the Bayesian Information Criterion (BIC)) to measure the fitness of a Bayesian network structure to the observed data, and it attempts to find the optimal model in the space of all possible Bayesian network structures. However, the conventional score-and-search approaches rely on efficient computation of the scoring function, which is only feasible for problems where data contain observations for all variables in the Bayesian network. Unfortunately, our domain has skill variables that are not directly observed. An intuitive work-around is to use EM to estimate the scoring function. However, in this case EM takes a large number (hundreds) of iterations that require Bayesian network inference, which is computationally prohibitive. Further, we need run EM for each candidate structure, and the number of possible Bayesian network structures is super-exponential with respect to the number of nodes. The Structural EM algorithm [7] is an efficient alternative.

Structural EM is an iterative algorithm that inputs a matrix $D$ of student performance (see example Table 1). Figure 2 illustrates one iteration of the Structural EM algorithm. The relevant steps are also sketched in Algorithm 1. Each iteration consists of an Expectation step (E-step) and a Maximization step (M-step). In the E-step, it first finds the maximum likelihood estimate $\Theta^*$ of the CPTs for the current structure $G$ calculated from previous iteration using parametric EM. It then does Bayesian inference to compute the expected values for the latent variables using the current model $(G, \Theta^*)$, and uses the values to complete the data. In the M-step, it uses the conventional score-and-search approach to optimize the structure according to the completed data (as if the latent variables were observed). Since the space of possible Bayesian network structures is super-exponential, exhaustive search is intractable and local search algorithms, such as greedy hill-climbing search, are often used. The E-step and M-step interleave and iterate until some stop criterion is met, e.g., the scoring function does not change significantly. Contrast to the conventional score-and-search algorithm, Structural EM runs EM only on one structure in each iteration, thus is computationally more efficient.

We use an efficient implementation of Structural EM available online called LibB\(^2\). Because COMMAND’s initialization step fixes the arcs from skills to items according to the $Q$-matrix, the M-step only needs to consider the candidate structures that comply with the $Q$-matrix. An advantage of using Structural EM to discover the prerequisite relationship of skills is that it can be easily extended to incorporate domain knowledge. For example, we can place constraints on the output structure to force or to disallow a skill to be a prerequisite of another skill. Another advantage of Structural EM is that it can be applied when there are missing data in the student performance matrix $D$ [7]. That is, some students do not answer all the items. The general idea is, in the E-step, the algorithm also computes the expected values for missing data points, in addition for latent variables.

3.3 Discriminate Between Equivalent BNs

Structural EM selects a Bayesian network model based on how well it explains the distribution of the data. Bayesian network theory states that some Bayesian networks are statistically equivalent in representing the data. Thus, the output from Structural EM is actually an equivalence class (EC) that may contain many Bayesian network structures\(^3\). These equivalent Bayesian networks have the same skeleton and the same v-structures\(^4\). For instance, Figure 3 gives an example of a simple equivalence class containing three Bayesian networks that are not distinguishable by Structural EM algorithm and the method in [14]. They share the skeleton but differ in the orientation of at least one of the edges (we will call such an edge a reversible edge). They apparently represent three different prerequisite structures.

3.3.1 Domain Knowledge

To determine a unique structure, we use a heuristic based in domain knowledge to determine the orientation of each reversible edge. For convenience in notation, let’s assume that the random variables that represent skill proficiency can take two values: $0$ if the skills is not mastered, and $1$ if the skill is mastered. Our assumption is that if a skill $S_1$ is the prerequisite of a skill $S_2$, a student can not master skill $S_2$ before she masters $S_1$. More formally:

**Assumption.** If $S_1$ is a prerequisite of $S_2$ (i.e., $S_1 \rightarrow S_2$), then $S_1 = 0 \Rightarrow S_2 = 0$. In other words, $P(S_2 = 0|S_1 = 0) = 1$.

Our assumption implies that $S_1$ cannot be a prerequisite of $S_2$ if $P(S_2 = 0|S_1 = 0) = 1$ does not hold. This puts a constraint on the joint distribution encoded by the Bayesian network to be learned.

For example, consider the case of choosing the orientation of a reversible edge $S_1 \rightarrow S_2$ from $S_1 \leftarrow S_2$ or $S_1 \rightarrow S_2$. We can check whether $P(S_2 = 0|S_1 = 0) = 1$ or $P(S_1 = 0|S_2 = 0) = 1$. However, it is possible that our assumption does not hold, and a student got to master a skill even if he does not know the prerequisite. Moreover, because of statistical noise, the conditional probability $P(S_2 = 0|S_1 = 0)$ may not be exactly 1. Thus, we use the following empirical rule:

\(^2\)http://compbio.cs.huji.ac.il/LibB/programs.html
\(^3\)Structural EM outputs a DAG. However, the scoring function does not discriminate between the many DAGs of the equivalence class.
\(^4\)A v-structure with nodes $u, v, w$ in a DAG are the directed edges $u \rightarrow v$ and $w \rightarrow v$ and $u$ and $w$ are not adjacent in the DAG [18].
Rule 1. If \( P(S_2 = 0|S_1 = 0) \geq P(S_1 = 0|S_2 = 0) \), we determine \( S_1 \rightarrow S_2 \); otherwise, we determine \( S_1 \leftarrow S_2 \).

Note that these two conditional probabilities can be computed easily from the Bayesian network model output from Structural EM. The intuition behind this rule is that the conditional probability \( P(S_2 = 0|S_1 = 0) \) can be interpreted as the strength of the prerequisite relationship \( S_1 \rightarrow S_2 \). The larger of this probability, the more likely the relationship \( S_1 \rightarrow S_2 \) holds. Since here we are concerned with which direction the edge goes, we simply compare the two probabilities and select the direction that is more probable. Note that \( P(S_2 = 0|S_1 = 0) = 1 \) and \( P(S_1 = 0|S_2 = 0) = 1 \) may hold simultaneously. If \( S_1 \rightarrow S_2 \) is true, \( P(S_1 = 0|S_2 = 0) = 1 \) only if \( P(S_2 = S_1 = 1) = 0 \) or if \( P(S_2 = S_1 = 0) = 1 \). If \( P(S_1 = 1) = 0 \), this implies that no student knows \( S_1 \). If \( P(S_2 = 0|S_1 = 1) = 0 \), it means that learning \( S_2 \) becomes trivial once students know \( S_1 \). For simplicity, we ignore this extreme case.

3.3.2 Theoretical Justification of Heuristic
We now provide theoretical justification for the rule we propose. Consider a simple equivalence class, which contains two equivalent DAGs \( S_1 \rightarrow S_2 \) and \( S_1 \leftarrow S_2 \), where the true model is \( S_1 \rightarrow S_2 \). We have three free conditional probability parameters: \( P(S_1 = 0) = p \), \( P(S_2 = 0|S_1 = 0) = q \), and \( P(S_2 = 1|S_1 = 1) = r \). Let’s define a ratio that quantifies choosing the true model:

\[
\text{ratio} = \frac{P(S_1 = 0|S_2 = 0)}{P(S_1 = 0)}
\]

Using Bayes rule and rules of probability, the rule \( \text{ratio} \geq 1 \) becomes \((1 − p)(1 − r) − p(1 − q) \geq 0\). Since \( \text{ratio} \) depends on \( p \), \( q \), and \( r \), we study how \( \text{ratio} \) changes with these parameters. Figure 4 shows the contour plots of \( \log(\text{ratio}) \) against \( P(S_1 = 0) \) and \( P(S_2 = 1|S_1 = 1) \) for three different values of \( P(S_2 = 0|S_1 = 0) \). The white region in each contour plot is the region where our heuristic fails because \( \text{ratio} < 1 \). Figure 4(a) shows that when \( P(S_2 = 0|S_1 = 0) = q = 1 \), our heuristic rule is always correct, no matter what, because there is no white space. With \( P(S_2 = 0|S_1 = 0) \) decreasing, the white region becomes larger and the rule becomes less accurate. As mentioned, \( P(S_2 = 0|S_1 = 0) \) can be interpreted as the strength of the prerequisite relationship. If we fix the value of \( P(S_2 = 0|S_1 = 0) \) and assume that the two free parameters \( p \) and \( r \) are independent and uniformly distributed, then the area of the white region represents the probability that the rule makes a wrong decision. As the strength of the prerequisite relationship becomes weaker, our rule to determine the prerequisite relationship becomes less accurate.

Figure 4: Contour plots of \( \log(\text{ratio}) \) against \( P(S_1 = 0) \) and \( P(S_2 = 1|S_1 = 1) \) for various values of \( P(S_2 = 0|S_1 = 0) \).

3.3.3 Orient All Reversible Edges
Using our proposed rule, we can orient every reversible edge in the network structure. However, orienting each reversible edge is not independent and may conflict with each other. Having oriented one edge would constrain the orientation of other reversible edges because we have to ensure the graph is a DAG and the equivalence property is not violated. For example, in Figure 5a, if we have determined \( S_1 \rightarrow S_2 \), the edge \( S_2 \rightarrow S_1 \) is enforced. In this paper, we take an ad-hoc strategy to determine the orientation for all reversible edges. For each reversible edge \( S_i \rightarrow S_j \), we let \( \text{ratio}^* = \frac{\text{ratio}}{\text{ratio}^*} \) if \( \text{ratio} \geq 1 \) and \( \text{ratio}^* = \frac{1}{\text{ratio}} \) otherwise. The larger the \( \text{ratio}^* \), the more confidently when we decide the orientation. We sort the list of reversible edges by \( \text{ratio}^* \) in descending order. We then orient the edges by this ordering. In our implementation, we use the following strategy: we first enumerate all equivalent Bayesian networks and make them a list of candidates; when an edge is oriented to \( S_i \rightarrow S_j \), we remove all contradicting Bayesian networks from the list. Eventually only one Bayesian network structure stands. This procedure is detailed in the Discriminate between equivalent BNs section of Algorithm 1. The EnumEquivalentDAGs(G) implements the algorithm of enumerating equivalent DAGs in [3].

4. EVALUATION
In §4.1, we evaluate COMMAND with simulated data to assess the quality of the discovered prerequisite structures. Then, in §4.2 we use data collected from real students. In all our experiments, we use BIC as the scoring function in Structural EM.

4.1 Simulated Data
Synthetic data allow us to study how COMMAND compares to the ground truth. For this, we engineered three prerequisite structures (DAGs), shown in Figure 5. Here, each figure represents different causal relations between the simulated latent skill variables.

Figure 5: Three different DAGs between latent skill variables. Item nodes are omitted.

For clarity, Figure 5 omits the item nodes: but each skill node is parent of six item variables and each item variable has 1-3 skill nodes as parents. All of these nodes are modeled using binary random variables. More precisely, the latent nodes represent whether the student achieves mastery of the skill, and the observed nodes indicate if the student answers the item correctly. Notice that these networks include the prerequisite structures as well as the skill-item mapping.

We consider simulated data with different number of observations (\( n = 150, 500, 1000, 2000 \)). For each sample size and each DAG, we generate ten different sets of conditional probability tables randomly with three constraints. First, we enforce that achieving mastery of the prerequisites of a skill will increase the likelihood of mastering the
skill. Second, for each prerequisite pair $S_i \rightarrow S_j$, $P(S_j = 0 | S_i = 0)$ is randomly selected to be in $\{0.9, 1.0\}$. Finally, mastery of a skill increases the probability of student correctly answering the test item. In total we generated 120 synthetic datasets (3 DAGs x 4 sample sizes x 10 CPTs), and report the average results.

We evaluate how well COMMAND can discover the true prerequisite structure using metrics designed to evaluate Bayesian networks structure discovery. In particular, we use the $F_1$ adjacency score and the $F_1$ orientation score. The adjacency score measures how well we can recover connections between nodes. It is a weighted average of the true positive adjacency rate and the true discovery adjacency rate. On the other hand, the orientation score measures how well we can recover the direction of the edges. It is calculated as a weighted average of the true positive orientation rate and true discovery orientation rate. In both cases, the $F_1$ score reaches its best value at 1 and worst at 0. Moreover, for comparison, we compute the $F_1$ adjacency score for Bayesian network structures whose skill nodes are fully connected with each other. These fully connected DAGs will serve as baselines for evaluating the adjacency discovery. For completeness, we list these formulas in tables 2 and 3, respectively.

$$F_1 = \frac{2 \cdot TPOR \cdot TDOR}{TPOR + TDOR}$$

Table 2: Formulas for measuring adjacency rate (AR)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>True positive (TPAR)</td>
<td>$\frac{# \text{ of correct adjacencies in learned model}}{# \text{ of adjacencies in true model}}$</td>
</tr>
<tr>
<td>True discovery (TDAR)</td>
<td>$\frac{# \text{ of correct adjacencies in learned model}}{# \text{ of adjacencies in true model}}$</td>
</tr>
<tr>
<td>$F_1$-AR</td>
<td>$\frac{2 \cdot TPAR \cdot TDAR}{TPAR + TDAR}$</td>
</tr>
</tbody>
</table>

Table 3: Formulas for measuring orientation rate (OR)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>True positive (TPOR)</td>
<td>$\frac{# \text{ of correctly directed edges in learned model}}{# \text{ of directed edges in true model}}$</td>
</tr>
<tr>
<td>True discovery (TDOR)</td>
<td>$\frac{# \text{ of correctly directed edges in learned model}}{# \text{ of directed edges in true model}}$</td>
</tr>
<tr>
<td>$F_1$-OR</td>
<td>$\frac{2 \cdot TPOR \cdot TDOR}{TPOR + TDOR}$</td>
</tr>
</tbody>
</table>

We use these metrics to evaluate the effect of varying the number of observations of the training set (sample size) on the quality of learning the prerequisite structure. We designed experiments to specifically answer the following four questions:

1. How does the type of items affect COMMAND’s ability to recover the prerequisite structure? We consider the situation where in the model each item requires only one skill and the situation where each item requires multiple skills.
2. How well does COMMAND perform when there is noise in the data? We focus on studying noise due to the presence of unaccounted latent variables.
3. How well does COMMAND perform when the student performance data have missing values?
4. How is COMMAND compared with other prerequisite discovery methods? In particular, we compare COMMAND to the Probabilistic Association Rules Mining (PARM) method [4].

We now investigate these questions.

6We do not compute $F_1$ orientation score for fully connected DAGs because all edges in a fully connected DAG are reversible.

4.1.1 Single-skill vs Multi-skill Items

We consider two situations where different types of Q-matrix are used. In the first situation, each item node maps to exactly one skill node. In the second one, each item maps to 1-3 skills. Figure 6 compares the $F_1$ of adjacency discovery and edge orientation results under the two types of Q-matrices. With only 500 observations, COMMAND improves on a fully connected Bayesian network baseline. COMMAND’s accuracy improves with the amount of data, but its accuracy is slightly lower when the Q-matrix contains items that require more than one skill. A possible explanation for this is that multi-skill items may introduce more spurious correlations in the data. With just 2000 observations, COMMAND recovers the true structures almost perfectly.

4.1.2 Sensitivity to Noise

Real-world data sets often contain various types of noise. For example, noise may occur due to latent variables that are not explicitly modeled. To evaluate the sensitivity of COMMAND to noise, we synthesize the three Bayesian networks in Figure 5 to include a StudentAbility node that takes three possible states (low/med/high). In these Bayesian networks, students’ performance depends not only on whether they have mastered the skills, but also on their individual ability. For simplicity, all items in the setting are single-skilled items. We first simulated data from Bayesian networks that have a StudentAbility variable to generate “noisy” data samples, and then use this data to recover the prerequisite structure. Figure 7 illustrates the procedure of this sensitivity analysis experiment for Structure 1.
We now evaluate COMMAND using two real-world data sets. We simulate data from Structure 3 from Figure 5(c) (with single-skill items), which has 21 pair-wise prerequisite relationships. We derive pair-wise prerequisite relationships from this network and see how the two approaches discover these relationships. When experimenting with PARM, we use \( \text{min} = 0.125, \text{minconf} = 0.76, \text{minprob} = 0.9 \), because they were suggested by the authors [4].

PARM is limited to discovering pair-wise prerequisite relationships (instead of constructing the full structure). To make a fair comparison, we evaluate how accurately COMMAND and PARM discover relationship pairs. For this, we use the \( F_1 \) metric in Table 2, but we count pairs of related skills instead of adjacencies. Two skills are related if one is a descendant of the other one. Figure 10 shows that COMMAND outperforms PARM, and the difference becomes significant for sample size \( n \geq 500 \). The low \( F_1 \) score of by PARM is because it fails to discover many prerequisite relationships (data not shown), and because PARM does not respect transitivity. For example, PARM may reject \( S_1 \rightarrow S_3 \) even if it has discovered \( S_1 \rightarrow S_2 \) and \( S_2 \rightarrow S_3 \). We speculate that selecting a different set of cutoff values for PARM may improve the results. However, determining these thresholds is not trivial and may require experts’ intervention. By contrast, COMMAND does not require tuning.

4.1.3 Sensitivity to Missing Values

Real-world datasets collected from students often have missing values, for example, when learners do not answer all items. To evaluate how COMMAND performs on data with missing values, we generated data sets of with 1000 observations with varying fraction of randomly missing values (10%, 20%, 30%, 40%, 50%). We used COMMAND to recover the structures from these data sets. Again, the models only contain single-skilled items. Figure 9 shows the results of this experiment. Although accuracy decreases when the fraction of missing values increases, COMMAND is able to recover the true structures for Structure 1 and 2 even when the data contain up to 30% missing values.

4.1.4 Comparison With Prior Work

The Probabilistic Association Rules Mining (PARM) is a recent algorithm for discovering the prerequisite relationships between skills [4]. In this approach, a prerequisite relationship \( S_1 \rightarrow S_2 \) is considered to exist if \( P(S_1 = 1, S_2 = 1) \geq \text{minsup} \land P(S_1 = 1 | S_2 = 1) \geq \text{minconf} \geq \text{minprob} \) and \( P(P(S_1 = 0, S_2 = 0) \geq \text{minsup} \land P(S_2 = 0 | S_1 = 0) \geq \text{minconf} \geq \text{minprob} \). We speculate that selecting a different set of cutoff values for PARM may improve the results. However, determining these thresholds is not trivial and may require experts’ intervention. By contrast, COMMAND does not require tuning.

4.2 Real Student Performance Data

We now evaluate COMMAND using two real-world data sets.

4.2.1 English Data Set

The Examination for the Certification of Proficiency in English (ECPE) dataset describes 2922 examinees in their understanding of English language grammar [16]. The dataset includes student performance in 28 items on 3 skills \( S_1 \): morphosyntactic rules, \( S_2 \): cohesive rules, and \( S_3 \): lexical rules). Each item requires either one or two of the three skills.

Figure 11 shows the prerequisite structure discovered with COMMAND. It hypothesizes that lexical rules is a prerequisite of cohesive rules and morphosyntactic rules; cohesive rules is a necessary skill for learning morphosyntactic rules. The pair-wise prerequisite relationships totally agrees with the findings in [16] and that by the PARM method in [4]. Our model infers a complete DAG, suggesting that there are no conditional independencies among the three
skills. This is an interesting insight that previous approaches cannot provide. Further, COMMAND also outputs the conditional probabilities associated with each skill and its direct prerequisite. We clearly see that the probability of student mastering a skill increases when the student has acquired more prerequisites of the skill.

### 4.2.2 Math Data Set

We now evaluate COMMAND using data collected from a commercial non-adaptive tutoring system. The textbook items are classified in chapters, sections, and objectives. We only use student performance data from tests in Chapter 2 and 3. That is, students are tested on the items after they have been taught all relevant skills.

**Q-matrix and preprocessing.** We define skills as book sections. We use a Q-matrix that assigns each exercise to a skill solely as the book section in which the item appears. For each chapter, we process the data to find a subset of items and students that do not have missing values. That is, the datasets we use in COMMAND have students responding to all of the items.

After filtering, two data sets, Math-chap2 and Math-chap3, were obtained for Chapter 2 and 3 respectively. In Math-chap2, six skills are included and each skill is tested on three to eight items, for a total of 30 items. In Math-chap3, seven skills are included and each skill has three to seven items, for a total of 33 items. Math-chap2 includes student test results for 1720 students, while the Math-chap3 has test results for 1245 students. For simplicity we use binary variables to encode performance data and skill variables.

**Prerequisite Structure Discovery.** The Bayesian networks generated with the COMMAND algorithm are illustrated in Figure 12. Our observation is that the topological order of the sections in both structures are fully consistent with the book ordering heuristic. This shows an agreement between our fully data-driven method and human experts. We also ran PARM approach to learn pair-wise prerequisite relationships from these data sets. Given $\text{minsup} = 0.125$, $\text{minconf} = 0.76$ and $\text{minprob} = 0.9$, 2, 5 $\rightarrow$ 2, 6, 2, 5 $\rightarrow$ 2, 7 and 2, 6 $\rightarrow$ 2, 7 are discovered for Math-chap2. 3, 1 $\rightarrow$ 3, 3 and 3, 2 $\rightarrow$ 3, 3 are discovered for Math-chap3. These relationships are small subset of the set of relationships discovered by COMMAND.

**Predictive Performance.** COMMAND outputs a Bayesian network model that can be used for inference and predictive modeling. For example, given a student’s response to a set of items, we can infer the student’s knowledge status of a skill. We could use COMMAND to identify students that may need remediation because they lack some background. We evaluate the accuracy of the predicted student performance on an item, when we observe the student response on the other items. More precisely, we compute the posterior probability of a student’s response to an item $I_i$ given his performance on all other items $I_{-i} = I \setminus \{I_i\}$, by marginalizing over the set of latent variables $S$:

$$P(I_i | I_{-i} = i_{-i}) = \sum_{S} P(I_i, S | I_{-i} = i_{-i}).$$

This probability can be computed efficiently using the Junction tree algorithm [11]. We then do binary classification based on the posterior probability to determine if the student is likely to answer correct. We compare the Bayesian network models generated from COMMAND with five baseline predictors:

- A **majority** classifier which always classifies an instance to the majority class. For example, if majority of the students get an item wrong, other students would likely get it wrong.
- A Bayesian network model in which the skill variables are **disconnected**. This model assumes that the skill variables are marginally independent of each other. Most existing knowledge tracing approaches make this assumption.
- A Bayesian network model in which the skill variables are connected in a **chain** structure, i.e., $2 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow \ldots$. This assumes that a section (skill) only depends on the previous section. In other words, a first-order Markov chain dependency structure.
- A Bayesian network model constructed using the pairwise relationships output from PARM. That is, we create an edge $S_i \rightarrow S_j$ if PARM says $S_i$ is the prerequisite of $S_j$. 

![Figure 11: The estimated DAG and CPTs of the ECPE data set.](image)

![Figure 12: Prerequisite structures constructed by COMMAND for Math data sets.](image)
A fully connected Bayesian network where skill variables are fully connected with each other. This model assumes no conditional independence between skill variables and can encode any joint distribution over the skill variables. However, it has exponential number of free parameters and thus can easily overfit the data.

The parameters of these baseline Bayesian network predictors are estimated from the data using parametric EM. The model predictions were evaluated using the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve metric calculated from 10-fold cross-validation. Results are presented in Figure 13. The error bars show the 95% confidence intervals calculated from the cross-validation. On both Math-chap2 and Math-chap3 data sets, the COMMAND models outperform the other five models. The fully connected models are the second best performing models. On Math-chap2, COMMAND model has an AUC of 0.803 ± 0.008 and the fully-connected model has an AUC of 0.791 ± 0.007 (Figure 13a). A paired t-test reveals that the AUC difference of two models are statistically significant with a p-value of 0.0022. On Math-chap3, COMMAND model has an AUC of 0.775 ± 0.007 and the fully-connected model has an AUC of 0.765 ± 0.008 (Figure 13b). The AUC difference of two models are also statistically significant with a p-value of 0.01. The fully connected models are outperformed by the much simpler prerequisite models, suggesting overfitting.

5. CONCLUSION AND DISCUSSION

Prerequisite graphs have been shown [1, 10] to improve student models. However, discovering the prerequisites between skills requires significant effort from subject matter experts. The main contribution of our work is a novel algorithm that simultaneously infers a prerequisite graph and a student model from data with less human intervention.

We extend on prior work in significant ways. We optimize the full structure of skills that captures the conditional independence between skills, instead of only estimating the pairwise relationships. Our experiments suggests that this results in better accuracy. Moreover, we argue that our strategy is easier to use because it does not require manual tuning of parameters. Other methods [2] require the guess and slip probabilities to be provided as input, or alternatively [4], thresholds to determine the existence of a prerequisite relationship. Determining these values requires experts’ intervention. COMMAND does not require such tuning.

We analyze how missing values, noise and dataset size can affect the performance of COMMAND. Further research could explore additional datasets and baselines. A secondary contribution of our work is that we develop a methodology to evaluate prerequisite structures on real student data. We believe that we are the first to compare prerequisite discovery strategies by how well they can be used to predict student performance. Therefore, we validate COMMAND not only with synthetic data, but with two real-world datasets. Our results suggest that COMMAND improves on the state of the art because it significantly improves on a recently published technique.

Learning a prerequisite graph is not merely discovering a Bayesian network—equivalent Bayesian network structures in fact represent different prerequisite structures. We believe we are the first to address this problem. We use domain knowledge to refine the prerequisite models output using a theoretically motivated method.

6. REFERENCES


Gauging MOOC Learners’ Adherence to the Designed Learning Path

Dan Davis∗, Guanliang Chen†, Claudia Hauff and Geert-Jan Houben
Delft University of Technology
Delft, the Netherlands
{d.j.davis, guanliang.chen, c.hauff, g.j.p.m.houben}@tudelft.nl

ABSTRACT
Massive Open Online Course (MOOC) platform designs, such as those of edX and Coursera, afford linear learning sequences by building scaffolded knowledge from activity to activity and from week to week. We consider those sequences to be the courses’ designed learning paths. But do learners actually adhere to these designed paths, or do they forge their own ways through the MOOCs? What are the implications of either following or not following the designed paths? Existing research has greatly emphasized, and succeeded in, automatically predicting MOOC learner success and learner dropout based on behavior patterns derived from MOOC learners’ data traces. However, those predictions do not directly translate into practicable information for course designers & instructors aiming to improve engagement and retention — the two major issues plaguing today’s MOOCs. In this work, we present a three-pronged approach to exploring MOOC data for novel learning path insights, thus enabling course instructors & designers to adapt a course’s design based on empirical evidence.

Keywords
MOOCs, learning path analysis, visualization

1. INTRODUCTION
MOOCs can deliver a world-class education on virtually any academic or professional development topic to any person with access to the Internet. Millions of people around the globe have signed up to courses offered on platforms such as edX, Coursera, FutureLearn and Udacity. At the same time though, only a very small percentage of these learners actually complete a MOOC successfully [15], an issue that continues to plague massive open online learning. Keeping MOOC learners engaged and improving the dismal retention rates are major concerns to instructional designers and MOOC instructors alike. Considerable research efforts have been dedicated to the automatic prediction of learners’ (imminent) dropout in MOOCs, e.g. [9, 12, 17, 24], under the assumption that once learners under the threat of attrition are identified, an automated intervention can be staged to (re)engage those learners with the course material. While the accuracy of these usually machine-learning-based predictors is high, their explanatory power is often low. Model features that have the strongest impact on prediction purely based on statistical grounds may not provide course designers & instructors with enough information to adapt the design or content of a MOOC in response.

In this work we aim to provide a more holistic view of learners’ progression through a MOOC in order to enable more practicable insights to instructors and designers. Our approach to educational data mining as presented here is a very literal realization of Graesser’s vision for the field by illustrating and “look[ing] at unique learning trajectories of individuals” [21]. We make use of the concept of learning paths (a learner’s route through course activities) and investigate how the learning paths of successful and unsuccessful MOOC learners differ.

The design of MOOCs on the edX platform1 implies a linear trajectory through the learning material. Most courses are broken up into weeks (Week 1, Week 2, etc.) and released one week at a time. Within these weeks, the standard instructional approach is to first provide a brief introduction to the week’s material, followed by the weekly video lectures (the main source of content delivery), then the assessments that evaluate learners’ knowledge of the preceding video lectures, and, finally, courses may offer bonus material. This cycle is repeated each course week (and sometimes multiple cycles comprise a single week). But do learners actually adhere to this cycle, and thus the designed learning path? Does it matter if they do not? These are the central issues that we focus on in this paper. While the concept of executed learning paths (i.e., the paths students actually take through a course) has received substantial attention in the e-learning and intelligent tutoring communities [13, 19], in the MOOC setting this concept has so far garnered little attention. First empirical evidence that learners do not always follow the designed sequence through a MOOC has been observed in [8], however, to our knowledge no in-depth investigation of this phenomenon in the MOOC context exists as of yet. We aim to close this knowledge gap and investigate the following.

1Our empirical work is based on edX MOOCs, but the same principles apply to other major MOOC platforms.

Proceedings of the 9th International Conference on Educational Data Mining

54
research question:
To what extent do learners adhere to a MOOC’s designed learning path?

We develop three approaches to characterize learning paths, thus providing three different views on a MOOC’s designed learning path (created by the course instructor or designer) and the executed paths (created by the learners of the MOOC). We apply our approaches on the log traces of more than 113,000 learners who participated in one of four edX-based MOOCs in the domains of computer science, political debates and business ethics. We show that (1) our approaches shed light on the deviations between designed and executed learning paths, and, (2) successful and unsuccessful learners differ considerably in the paths they follow. We believe that our work can provide instructional designers a valuable analysis tool to improve the design of both online courses and MOOC platforms in the future as they provide data-driven insights into the actual behavior of learners and the impact of their behaviors on learning outcomes.

2. RELATED WORK

In this section, we elaborate on existing research in learner modeling [5], focusing on works that investigate learning activity sequences and their impact on learning outcomes. The problem solving behavior of learners in the context of e-learning and intelligent tutoring systems has been explored in [10, 13, 14, 19]. In contrast to our work, which considers a range of activities learners perform throughout a course (and compares them to the designed learning path), these works have explored learners’ exhibited behavior within only one activity type — problem solving. Specifically, Köck and Paramythitis [14] performed activity sequence clustering (an application of sequential pattern mining [22]) to model the learners’ behavior, while in [13] automated clustering and human synthesis of the generated clusters were combined to identify patterns of problem solving. Shanabrook et al. [19] introduced a semi-automatic approach to identify a student’s state while problem solving (including: gaming the system, guessing out of frustration, abusing hints, being on-task) in a high school-level intelligent tutoring system employing sequence-based motif discovery. Jeong and Biswas [10] developed a Hidden Markov Model to describe how different middle school student behavior trends lead to different learning processes & outcomes when problem solving. In the context of MOOCs, sequences of learning activities have been explored by Wen and Rosé [23], who investigated the most common two-step activity sequences learners exhibit across two MOOCs. These patterns were then manually checked and analysed for interesting learning habits. A similar analysis of two-step chains was performed in Guo and Reinecke [8] who found that learners generally progress through the course content in a non-linear, “exploratory” manner [16]. Guo and Reinecke [8]’s observation of learners frequently performing “backjumps” (moving from a quiz to a lecture video previously introduced) can be considered as one of the first comparisons of executed and designed learning paths in MOOCs. Kızıllec et al. [11] (replicated in [6]) have also taken steps in this direction, by utilizing the assessment submission times (either on track, late or never) in MOOCs as indicators of learner engagement groups (completing, auditing, disengaging or sampling learners). Our work can be considered a significant expansion to these approaches, as we explore longer activity sequences (eight-step chains), thus enabling the discovery of more high-level and complex patterns and making designed vs. executed paths the focal point of our investigation.

Video interactions in MOOCs were the focus of Sinha et al. [20], who categorized the most prominent chains of video interactions (pause, play, speed, and skipping) and analyzed them with respect to learner dropout. MOOC discussion patterns have been investigated by Brooks et al. [3] who found that MOOC students exhibit markedly different discussion patterns than were expected based on blended learning environments. This finding can also be considered as a motivation for our work; MOOCs may not always be used by learners the way the instructors or course designers intended. The concepts of process mining and conformance checking, in particular, are also employed in areas such as business process execution; [18] explains how business processes can be monitored (process mining) and then compared to the intended model (conformance checking) via a measure of fitness.

3. SUBJECTS & DATA

We explore our research question in the context of four MOOCs: Functional Programming (teaching the functional programming paradigm), Data Analysis (teaching spreadsheet and basic Python skills for data analysis), Framing (the art of political debates), and Responsible Innovation (a MOOC on the ethics and safety of new technologies). All MOOCs were offered on the edX platform in 2014/2015 and designed as xMOOCs.

Overview of MOOCs. Table 2 provides an overview of the four MOOCs in this study. The learner enrollment varies between ≈9k and ≈37k. While the four MOOCs are comparable in their video material offerings (between 41 and 59 videos), they differ significantly in the number of summative assessment questions (between 26 and 288 quiz questions). We also observe considerable differences in the percentage of video material watched by certificate-earning learners (replacing [8]) — less than half of the videos are accessed by successful learners in Data Analysis, while more than two thirds of the videos are accessed by successful learners in Functional Programming. Lastly, we note that the Responsible Innovation MOOC is an outlier with respect to the percentage of learners that passed the course without streaming any video material, 2 with nearly 20% of successful learners falling into this category; the same applies for only ≈4% of learners in the other three MOOCs.

Translating Log Traces into a Semantic Event Space. The edX platform provides a great deal of timestamped log traces, including clicks, views, quiz attempts, and forum interactions. We adapted the MOOCdb toolkit to our needs and translated these low-level log traces into a data schema that is easily query-able.

For this work, we focus on four event types as listed in Table 2: events related to videos, quizzes, progress pages, and discussion forums. Videos can be watched - this event

4Note that the log traces did not capture video downloads and subsequent offline watching.
3http://moocdb.csail.mit.edu/
Table 1: Overview of the MOOCs in our study. The #Chains column contains the number of events observed throughout the MOOC (cf. Table 2). The “Passing Grade” shows the percentage of quiz questions to answer correctly to receive a course certificate. “Tries” indicates how many attempts a learner has per question. “Videos Accessed” shows the average % of course videos watched by certificate-earning learners. “Missing” is the % of certificate-earning learners who streamed zero video lectures.

<table>
<thead>
<tr>
<th>MOOC</th>
<th>Enrolled</th>
<th>Pass Rate</th>
<th>Chains/Chains Weeks</th>
<th>Videos</th>
<th>Quiz Questions</th>
<th>Passing Grade</th>
<th>Tries</th>
<th>Videos Accessed</th>
<th>Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional</td>
<td>37,485</td>
<td>5.3%</td>
<td>1.06M/807k 14</td>
<td>41</td>
<td>288</td>
<td>60%</td>
<td>1</td>
<td>67.5%</td>
<td>4.3%</td>
</tr>
<tr>
<td>Programming</td>
<td>8,850</td>
<td>4.3%</td>
<td>66k/30k 7</td>
<td>47</td>
<td>75</td>
<td>50%</td>
<td>1-3</td>
<td>49.7%</td>
<td>19.6%</td>
</tr>
<tr>
<td>Responsible</td>
<td>34,017</td>
<td>2.4%</td>
<td>95k/141k 6</td>
<td>55</td>
<td>26</td>
<td>50%</td>
<td>2</td>
<td>51%</td>
<td>3.8%</td>
</tr>
<tr>
<td>Innovation</td>
<td>33,515</td>
<td>6.5%</td>
<td>1.02M/855k 8</td>
<td>59</td>
<td>136</td>
<td>60%</td>
<td>2</td>
<td>45%</td>
<td>3.6%</td>
</tr>
<tr>
<td>Data Analysis</td>
<td>34,017</td>
<td>2.4%</td>
<td>95k/141k 6</td>
<td>55</td>
<td>26</td>
<td>50%</td>
<td>2</td>
<td>51%</td>
<td>3.8%</td>
</tr>
</tbody>
</table>

Table 2: Overview of events considered in this work.

<table>
<thead>
<tr>
<th>Video</th>
<th>Quiz</th>
<th>Progress</th>
<th>Forum</th>
</tr>
</thead>
<tbody>
<tr>
<td>WATCH</td>
<td>START</td>
<td>VIEW</td>
<td>START</td>
</tr>
</tbody>
</table>

is generated whenever a user clicks the video ‘play’ button. Quizzes are identified through the beginning of the quiz session (the user enters the quiz page), the submission of one or more answers, and the ending of the quiz session (the user leaves the quiz page). Those quizzes are typically summative in nature. If a user views his or her progress page, the VIEW event is elicited. Finally, we condense discussion forum events into three kinds of items: the start of a forum session (the user first enters the forum), the submission of content (question, comment or reply) and the end of the forum session (the user leaves the forum page).

All executed learning paths that we extract from the learner log traces consist of the events listed in Table 2. The rationale for choosing these events comes from the designed learning path by which xMOOCs are typically formed: first watch one or more lecture videos, and then move on towards the quiz and/or forum section for assessment and knowledge building & verification respectively. In Figure 2 we visualize a week's designed learning path for each of the four MOOCs we study (this pattern is repeated in every course week). Video lectures form a common denominator, starting the path. Functional Programming and Data Analysis rely on videos and quizzes only (with Data Analysis exhibiting multiple video-quiz “cycles” within a week), whereas Responsible Innovation and Framing make use of the forums as well. The learning path shown for Framing does not include quizzes as they are posed only in the final week (in the form of an exam).

4. APPROACH

Having introduced the subjects of our work and the events we consider, we now describe the three distinct approaches to the visualization & exploration of executed learning paths (that is, learners’ sequential movement over time through the activities offered in a MOOC) we developed.

4.1 Video Interactions

As shown in Figure 2, videos are a focal point of xMOOCs. Accordingly, in a first analysis, we focus exclusively on video interactions and explore to what extent learners adhere to the designed video watching learning path. Therefore, in this study we only make use of WATCH events.

We transform the WATCH events generated by a set of learners into a directed graph $G_{M,L} = (V_M, E_{M,L})$ - as the subscripts indicate, with $M$ fixed, the set $V$ is independent of the subset of learners chosen, while $E$ is dependent on the learners in $L$. All lecture videos contained in $M$ form the set of vertices $V_M$. The vertices are labelled chronologically, that is, for any vertex pair $(v_i, v_j)$ with $i < j$, the corresponding lecture video $i$ must appear in the designed learning path before video $j$. The edges are directed and weighted according to the number of WATCH events by the learners $L$: an edge between $v_i$ and $v_j$ presents the learners’ transition between these videos, i.e. the number of times learners watching video $v_i$ watch $v_j$ next, before any other video. To discover whether or not there are marked differences in the way different groups of learners behave, we generate the video interaction graph for different sets of learners, such as successful (certificate earning) vs. unsuccessful learners.

4.2 Behavior Pattern Chains

Having considered the transitions between lecture videos, we now turn to the exploration of transition patterns among all eight events identified in Table 2. Previous works [23] have viewed MOOC learner patterns either in terms of one-step directed pairs of events (such as watch video $\rightarrow$ begin quiz) or based on video click chains only [20]. One-step chains can only provide limited insights into more high-level behavioral patterns — we may, for instance, be in-
interested to understand how many learners are “binge watchers” (watching many videos in a row) or “strategic learners” (looking at quiz questions before watching the corresponding lecture video). In order to contribute insights to our research question we need to consider longer chains. We have settled on eight-step chains, as they provide insights into more high-level concepts but are still numerous enough in our log traces to make claims about their general usage. We consider all events of Table 2 and create event chains by sliding a window of size eight over each learner’s chronologically ordered learning path through a MOOC. An example eight-step chain this procedure yields is shown in Figure 1. To identify the underlying trends in the chains, we employed the open card sort approach [7]. After printing out two sets of the thirty most frequently occurring chains on paper, two authors independently sorted them into (non-predefined) like-groups by hand and afterwards discuss the differences in each sort, creating a composite of the two results. The outcome of this method is a synthesis of similar chain types into groups sharing the same motif, or recurring theme. Based on the motifs, we created a rule–based system that assigned a MOOC’s entire set of chains to the identified motifs (chains that do not fit into any motif are left “unassigned”). This process is repeated for each of the MOOCs we investigate. The advantage of this approach over the automatic clustering of the chains is the infusion of our domain knowledge into the clustering process.

### 4.3 Event Type Transitions

Lastly, we explore event type transitions, or how likely learners are to move from one event type to another. Inspired by the methods employed in [10, 13, 14] we use discrete-time Markov chains (a memory-less state transitioning process encoding how often learners move from one event type to another) in order to chart the likelihood that a learner will transition from one engagement activity to another. Whereas the prior works employ these methods in the context of problem solving (knowledge assessment), we focus on the larger process of knowledge building, which transpires over the span of an entire course. While it may be self-evident that non-passing learners answer less quiz questions than their certificate-earning peers (and thus the transition probabilities to $\text{SUBMIT}_{\text{QUIZ}}$ are likely to be lower for non-passers), the visualization of the Markov chains enables designers to pinpoint the differences in transitions between different types of learners (e.g. passers vs. non-passers) across all events in one coherent plot.

![Figure 1: An example eight-step chain.](image1)

![Figure 2: The designed learning path for a standard week (Week 4) of each MOOC. The circled numbers indicate the step number of each transition in that week’s sequence. Notice the diversity in course designs that characterize these four MOOCs.](image2)
5. FINDINGS
To answer our research question (do learners adhere to the designed learning path?), we apply the three approaches outlined in Section 4 to the datasets described in Section 3.

5.1 Video Interactions
We visualize the video interactions across the first three weeks (these are where the most deviations occur; the later weeks are more in line with the designed path) of each MOOC in Figures 3 to 6, distinguishing two sets of learners: those that eventually earn a certificate (“Passing”) and those that do not (“Non-Passing”). The designed video interaction learning path is exhibited by the left-to-right flow of the vertices (one per video). The edges correspond to the executed learning paths — with edge thickness indicating the (normalized) number of learners having taken that path (only the 90% most frequently occurring transitions each week are shown); the set of red edges represent the executed transitions that follow the designed transitions. A number of observations can be made based on the visualizations: (i) passing learners deviate considerably less from the designed learning path than non-passing learners across all four MOOCs, (ii) passing learners are more likely to skip video lectures introducing the platform (the first three videos in the Framing MOOC) than non-passing learners, indicating a higher level of seniority in MOOC-taking, (iii) towards the end of week three, the deviations among the sets of passing and non-passing learners are negligible (i.e. the non-passing learners still active exhibit a similar video watching behavior as the passers), and (iv) skipping videos — jumping ahead — is much more common than backtracking — jumping backwards — for both passers and non-passers.

An emerging object in the field of Design (and gaining some attention in the field of Software Design [4]) is that of desire paths, or paths not intended by the designer, but those which “arise due to off-path use ... for a variety of purposes such as access to places of interest and shortcutting” [2]. This research serves as a reminder that desire paths indeed exist in MOOCs (as evident in the skipping of introductory lecture material) — they just have not yet been made as visible as those brown stripes of beaten grass and dirt transecting public parks and trails. They are a reminder that humans can collectively communicate good design by their actions.

5.2 Behavior Pattern Chains
Our second approach explores learners’ behavioral patterns. As outlined in Section 4.2, we first manually clustered and labelled the most frequent eight-step pattern chains in order to determine what type of behaviors (or motifs) learners exhibit beyond a single-click transition, before automatically assigning the remaining chains into those motifs. Depending on the MOOC, this approach yielded between eight and 11 motifs, with some motifs appearing only in a subset of courses. For brevity reasons, in Tables 3 to 6 for each MOOC we list its most frequent motifs (specifically those into which ≥2% of all chains are classified); as a comparison in Table 3 we also list the total number of chains generated by passing/non-passing learners in each MOOC — depending on the MOOC, the listed motifs capture between 42%–77% of the total number of chains. Whenever a motif first introduced, we briefly describe which event types and event orderings characterize it.

Examining the results, we observe that (i) Binge Watching is a frequent motif in all MOOCs with non-passers always exhibiting more binge watching (i.e. watching videos uninterrupted by other activities) than passers, (ii) the Lecture→Quiz Check motif, which captures the “classic” xMOOC idea of video watching with subsequent question answering is frequent in three of the four MOOCs, however no consistent divergent behavior for passers and non-passers is found, (iii) motifs with forum events occur in three of the four MOOCs — by course design in Framing and Responsible Innovation (cf. Figure 2), but not in Functional Programming, indicating issues related to material clarity, and (iv) the Quiz Check motif, which is exhibited by learners checking the quiz questions without answering any of them (which is usually followed by video watching and subsequent quiz completion), is only found in one MOOC frequently; in Data Analysis 2% of the chains follow this motif, a smaller percentage than we expected, indicating that very few learners are gaming the system by “attempting to succeed in an educational environment by exploiting properties (quiz ques-

Note, that we implemented our rules for the automatic assignment of chains to motifs according to these characterizations.

It does not appear among the frequent motifs in Framing, which has a final exam instead of weekly quizzes.
tions are posted alongside the video material) of the system (edX platform) rather than by learning the material and trying to use that knowledge to answer correctly," [1].

Table 4: Most frequent motifs (≥2% chains) in Functional Programming.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Quiz Complete</td>
<td>552,363</td>
<td>328,905</td>
<td>223,368</td>
</tr>
<tr>
<td>2 Binge Watching</td>
<td>149,784</td>
<td>59,498</td>
<td>90,286</td>
</tr>
<tr>
<td>3 Lecture→Quiz Complete</td>
<td>100,179</td>
<td>50,415</td>
<td>49,764</td>
</tr>
<tr>
<td>4 Quiz Complete→Forum</td>
<td>99,828</td>
<td>67,722</td>
<td>32,106</td>
</tr>
<tr>
<td>5 Quiz Complete→Progress</td>
<td>38,854</td>
<td>26,126</td>
<td>12,728</td>
</tr>
</tbody>
</table>

Table 3: Most frequent motifs (≥2% chains) in Functional Programming.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Quiz Complete</td>
<td>18,446</td>
<td>11,377</td>
<td>7,069</td>
</tr>
<tr>
<td>2 Binge Watching</td>
<td>12,530</td>
<td>8,461</td>
<td>4,069</td>
</tr>
<tr>
<td>3 Lecture→Quiz Complete</td>
<td>9,060</td>
<td>3,752</td>
<td>1,308</td>
</tr>
<tr>
<td>4 Lecture→Forum→Lecture</td>
<td>3,910</td>
<td>2,386</td>
<td>1,524</td>
</tr>
<tr>
<td>5 Quiz Complete→Progress</td>
<td>3,741</td>
<td>2,898</td>
<td>843</td>
</tr>
</tbody>
</table>

Table 5: Most frequent motifs (≥2% chains) in Framing.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Quiz Complete</td>
<td>169,786</td>
<td>116,878</td>
<td>52,908</td>
</tr>
<tr>
<td>2 Quiz Complete→Lecture→Quiz</td>
<td>145,596</td>
<td>82,247</td>
<td>63,349</td>
</tr>
<tr>
<td>3 Binge Watching</td>
<td>87,760</td>
<td>28,066</td>
<td>59,694</td>
</tr>
<tr>
<td>4 Lecture→Quiz Complete</td>
<td>78,790</td>
<td>41,543</td>
<td>37,247</td>
</tr>
<tr>
<td>5 Quiz Complete→Lecture</td>
<td>43,612</td>
<td>21,916</td>
<td>21,696</td>
</tr>
<tr>
<td>6 Quiz Check</td>
<td>37,406</td>
<td>19,444</td>
<td>17,962</td>
</tr>
</tbody>
</table>

Table 6: Most frequent motifs (≥2% chains) in Data Analysis.

Analysis learners mostly focus their attention on lectures and assessments, with little concern for discussion. The visualizations also reveal at which specific moments learners seek feedback on their progress (i.e. make a transition to the Progress vertex), such as after a Quiz or Forum in Responsible Innovation and Framing. These movements are not included in any of the courses’ designed paths; course designers can use this insight to proactively insert feedback in order to encourage more awareness and self-regulated learning. When comparing transitions of passing vs. non-passing learners, we observe that (i) non-passers make the transition to the video event from more diverse event types than passers (indicating that non-passers’ executed paths follow the designed path to a lesser degree than passers’ executed paths), (ii) video-to-video transitions are more prevalent among non-passers (in line with our findings on the binge watching motif), and (iii) passing learners are more likely to move from Quiz Start to Quiz Submit, while non-passing learners are more likely to move from Quiz Start to Quiz End (without answering a question).
6. CONCLUSION
Before adaptive learning systems can reach their potential, two important baselines must be established: (i) the precise learning path the instructor wants the student to follow and (ii) students’ natural behavior within the course. Adaptive instruction will be most effective when the differences between these two baselines are both identified and addressed. The present research offers novel insights into the identification of those differences.
Specifically, in this work we have introduced three different approaches (the video interaction graph, behavior pattern chains and event type transitions) to explore and visualize MOOC log traces with respect to the designed and executed learning paths.
We have applied our approaches on the log traces of four different edX-based MOOCs (from different domains and different pedagogical structures) and have shown to what extent learners (as a whole group as well as partitioned into passing and non-passing learners) follow the prescribed path. In future work, we will expand our analyses to a larger set of MOOCs to gain a greater understanding of the “classes” of xMOOCs that exist on the major MOOC platforms today. We also plan to consider more diverse sub-populations of learners in future analyses, beyond passing or not passing. We will also investigate semi-automatic approaches to the adaptation of MOOC learning paths, in order to minimize the gap between designed and executed paths as well as the impact this work has on engagement, retention, learner success and more fine-grained learner partitions (such as completing, auditing, and sampling learners [11]).

References
Figure 7: Markov Model state visualization of non-passing (left) and passing (right) learners in the Data Analysis MOOC. Edges with weights below 20% are hidden from view.

Figure 8: Markov Model state visualization of non-passing (left) and passing (right) learners in the Functional Programming MOOC. Edges with weights below 20% are hidden from view.

Figure 9: Markov Model state visualization of non-passing (left) and passing (right) learners in the Framing MOOC. Edges with weights below 20% are hidden from view.

Figure 10: Markov Model state visualization of non-passing (left) and passing (right) learners in the Responsible Innovation MOOC. Edges with weights below 20% are hidden from view.
Dynamics of Peer Grading: An Empirical Study

Luca de Alfaro
University of California, Santa Cruz
Department of Computer Science
luca@ucsc.edu

Michael Shavlovsky
University of California, Santa Cruz
Department of Computer Science
mshavlov@soe.ucsc.edu

ABSTRACT
Peer grading is widely used in MOOCs and in standard university settings. The quality of grades obtained via peer grading is essential for the educational process. In this work, we study the factors that influence errors in peer grading. We analyze 288 assignments with 25,633 submissions and 113,169 reviews conducted with CrowdGrader, a web based peer grading tool. First, we found that large grading errors are generally more closely correlated with hard-to-grade submission, rather than with imprecise students. Second, we detected a weak correlation between review accuracy and student proficiency, as measured by the quality of the student’s own work. Third, we found little correlation between review accuracy and the time it took to perform the review, or how late in the review period the review was performed. Finally, we found a clear evidence of tit-for-tat behavior when students give feedback on the reviews they received. We conclude with remarks on how these data can lead to improvements in peer-grading tools.

1. INTRODUCTION
In peer grading, students review and grade each other’s work. The grades assigned by the students to each item are then merged into a single consensus grade for the item. Peer grading has several benefits, as reported in the literature, including the fact that students learn from each other’s work, and the reduced workload on the instructors. For these reasons, peer grading has been widely used both in MOOCs, where it would be infeasible for a small number of instructors to grade all work [14, 1, 5, 12], and in standard university classes [17, 15, 10, 18, 3, 16].

Successful peer grading is predicated on the ability to reconstruct a reasonably accurate consensus grade from the grades assigned by the students. This leads to the following question: what factors cause or influence the errors in peer-assigned grades? We are interested in this question for three reasons. First, we wish to obtain a better understanding of the dynamics and human factors in peer grading. Second, a better understanding of the causes of error has the potential to lead to tool improvements that reduce the errors. For example, if mis-understanding on the work submitted constituted a large source of error, then peer grading tools could be augmented with means for work authors and graders to communicate, so that the misunderstandings could be resolved. Third, a better model of peer grading errors might lead to better algorithms for aggregating the student-assigned grades into the consensus grades for each item.

Our interest in the origin of peer-grading errors is also due to our work on the peer-grading tool CrowdGrader [8]. We have put considerable effort in reducing the error in the consensus grade computed by CrowdGrader, as compared to control instructor-assigned grades. While efforts on the tool UI and UX paid off, as we will detail later, the efforts to create more precise grade-aggregation algorithms did not. In the context of MOOCs, [14] reports a 30% decrease in error using parameter-estimation algorithms that infer, and correct for, the imprecision and biases of individual users. CrowdGrader is used mostly in universities and high-schools. On CrowdGrader data, the parameter-estimation algorithm of [14] offers no benefit compared with the simple “Olympic average” obtained by removing lowest and highest grades, and averaging the rest. Indeed, we have spent a large amount of time experimenting with variations upon the algorithm (see also [7]) and new ideas, but we are yet to find an algorithm that offers consistent error reduction of more than 10% compared to the Olympic average. Thus our interest on the origin of errors in CrowdGrader: what are the main causes? What makes them so difficult to remove using algorithms based on parameter estimation, reputation systems, and more?

To gain an understanding of the dynamics of peer grading, we have analyzed a set of CrowdGrader data consisting in 288 assignments, 25,633 submissions, and 113,169 grades and reviews. Of the 25,633 submissions, 2,564 were graded by the instructors in addition to the students. The questions we ask include the following.

Is error mostly due to items or to students? We first ask the question of whether the imprecision in peer grades can be best explained in terms of students being imprecise, or items being difficult to grade. We answer this question in two different ways.

First, we build a parameterized probabilistic model of the review process, similar to the model of [14], in which every review error is the sum of a component due to the submission being reviewed, and of a component due to the reviewer. The parameters of the model are then estimated via Gibbs sampling [11]. The results indicate that students contribute roughly two thirds of the total evaluation error.
This result, however, speaks to the average source of error. Of particular concern in peer grading are the very large errors that happen less frequently, but have more impact on the perceived fairness and effectiveness of peer grading. We measure the correlation of large errors in items, and in users; our results indicate that hard-to-grade items are a more common cause of large errors than very imprecise students.

Do better students make better graders? A natural question is whether better students make better graders. In Section 6 we give an affirmative answer: students whose submissions are in the lower 30%-percentile quality-wise have a grading error that is about 15% above average. The effect is fairly weak, a likely testament to the fundamental homogeneity in abilities in a high-school or college class, as well as to the fact that grading a homework is usually easier than solving the homework.

Does the timing of reviews affect their precision? In Section 7 we consider the relation of review timing and review precision. We did not detect strong dependencies between grading error and the time taken to complete a review, the order in which the student completed the reviews, or how late the reviews were completed with respect to the review deadline.

Does error vary with class topic? In Section 4 we consider the question of whether grading precision varies from topic to topic. Comparing broad topic areas, such as computer science, essays, science, we find the statistics to be quite similar, indicating how general factors are less important than the specifics of each class.

Does tit-for-tat affect review feedback? CrowdGrader allows students to leave feedback on the reviews and grades they receive; this feedback is then used as one of the factor that determines the student’s grade in the assignment. The feedback was introduced to provide an incentive for writing helpful reviews. In Section 8 we show that when a grade is over 20% below the consensus, it receives a low feedback score due to tit-for-tat about 38% of the time.

In the next section, we give a brief description of CrowdGrader, and of the datasets on which our analysis is based. The subsequent sections present the details of the answers to the above questions. We conclude with a discussion on the nature of errors in peer grading, and on the implications for algorithms and reputation systems for computing consensus grades.

2. RELATED WORK

The accuracy of peer grading in the context of MOOCs has been analyzed in [13], where the match between instructor grade and student grades is analyzed in detail. The study finds a tendency by student to rate higher people that share their country of origin — and this in spite of the grading process being anonymous. The study finds that improvement in grading rubrics lead to improved grading accuracy. Geographical origin, along with gender, employment status, and other factors, are found to have influence on engagement in peer grading in a French MOOC in [4]. Our work is thus somewhat orthogonal to [4, 13]: we do not have data on student ethnicity, and we focus instead on factors measurable from the peer grading activity itself.

Frequently, peer grades are accompanied with reviewers’ comments or feedback; [19] explores the possibility of using the review text to assess review quality. The authors show a successful application of classifiers and statistical Natural Language Processing to evaluate reviews.

Peer Instruction is a process in which students can observe grades by other reviewers, discuss the review, and consequently modify their grades [6]. The factors that influence grades in peer instruction have been studied in [2]. In spite of the different settings, [2] also observe that the behavior of high and low-scoring students is fairly similar in terms of their grading accuracy.

3. THE CROWDGRADER DATASET

To analyze the source of grading errors in peer grading, we rely on a dataset from CrowdGrader, a peer review and grading tool used in universities and high-schools [8]. After students submit their solutions to an assignment, students review and grade a certain number of submissions by their peers. From these peer grades, CrowdGrader computes a consensus grade for every submission. Once the review phase is concluded, the students can rate the reviews they received according to a 1 to 5-star rating. These review ratings are meant to provide an incentive for students to write detailed, helpful reviews of other students work.

The overall dataset we examined consisted in 288 assignments, for a total of 25,633 submissions and 113,169 reviews, written by 23,762 distinct reviewers. The number of reviewers is smaller than the number of submissions, as some students did not participate in the review phase. Table 1 gives a break-down of the dataset according to subject area. On average, each submission received 4.41 reviews, and each reviewer wrote on average 4.76 reviews.

We will refer to submissions also as items, and we will refer to students or reviewers also as users, thus adopting common terminology for general peer-review systems. CrowdGrader includes three features that promote grading accuracy; these features likely influenced the data presented in this study.

Incentives for accuracy. The overall grade a student receives in a CrowdGrader assignment is a weighed average of the student’s submission, accuracy, and helpfulness grades. The accuracy grade reflects the precision of the student’s grade, compared either to the other grades for the same submission or, when available, to the instructor-assigned grade. The helpfulness grade reflects the rating received by the reviews written by the student. Combining the submission grade with the accuracy grade creates an incentive for students to be precise in their grading. The amount of incentive can be chosen by the instructor, but the default is to give 75% weight to the submission grade, 15% weight to the accuracy grade, and 10% weight to the helpfulness grade, and most instructors do not change this default.

Ability to decline reviews. Early in the development of CrowdGrader, we noticed that some of the most glaring grading errors occurred when reviewers were forced to enter a grade for submissions that they could not properly evaluate. This occurred, for instance, when students could not open the files uploaded as part of the submission, due to software incompatibilities. To mitigate this problem, we gave students the ability to decline to perform reviews of particular submissions. The total number of submissions a student can decline is bounded, to prevent students from “shopping around” for the easiest submissions to review.

Submission discussion forums. Another early source of large errors
in CrowdGrader consisted in gross mis-understandings between the author of a submission, and the reviewers. For instance, when zip archives are submitted, the reviewers may expect some information to be contained in one of the component files, whereas the author might have included it in another. Another example consists in mis-organizing the content of a software submission, so that the reviewers do not know how to run it and evaluate it. To remedy this, CrowdGrader introduced anonymous forums associated with each submission, where submission authors and reviewers can discuss any issues they encounter in evaluating the work.

4. ERRORS IN PEER GRADING

Instructor grades and Olympic averages. We measure review error as the difference between individual student grades, and the "consensus grade" for each submission. We consider two kinds of consensus grades. One is the Olympic average of the grades provided by the students; this is obtained by discarding the lowest and highest grade for each submission, and taking the average of the remaining grades. The other is the instructor grade. In CrowdGrader, instructors (or teaching assistants) have the option of regrading submissions. In some assignments, instructors decided to grade most submissions as control; in other assignments, instructors mostly re-graded only submissions where student grades were in too much disagreement. When considering instructor grades, we consider only assignments of the first type, where instructors graded at least 30% of all submissions. Considering assignments where instructors grade only problematic submissions would considerably skew the statistics. The dataset, for instructor grades, is thus reduced to 19 assignments and 7675 reviews. Instructor grades and Olympic averages.

Global and per-topic errors. Table 2 reports the size of errors in CrowdGrader peer grading assignments, split by assignment topic, and taking instructor grades and Olympic grades as reference. When the error is measured with respect to instructor grades, computer science, physics, and high-school assignments showed smaller average error than business, sociology and English, all of whose assignments required essay-writing. When the error is measured with respect to Olympic average, is is mainly business and English that show larger error.

5. ITEM VS. STUDENT ERROR

We consider in this section the question of whether error can be attributed predominantly to imprecise students, or to items that are difficult to grade.
parameters $\alpha_0, \beta_0, \alpha_1, \beta_1, \mu_0, \gamma_0$. In order to apply Gibbs sampling, we need to start from suitable prior values for the quantities being estimated. To obtain suitable priors for the distribution of item quality, we first compute an estimated grade for each item using Olympic average, and we obtain $\mu_0$ and $\gamma_0$ by fitting a normal distribution to the estimated grades. To estimate prior parameters $\alpha_0, \beta_0$ of student reliabilities we fit a Gamma distribution to a set of approximated students reliabilities. In detail, for every student $u$ we populate a list of errors $l_u$ by the student. Again, we computer errors with respect to the average item grades after removing the extremes (the Olympic average). Using the list of error $l_u$, we estimate a standard deviation $\sigma_u$ for every student $u \in U$. This allows us to approximate student reliability $\tilde{\tau}_u$ as $\frac{1}{\sigma_u}$. Prior parameters $\alpha_0, \beta_0$ are obtained by fitting a Gamma distribution to the set of estimated student reliabilities $\{\tilde{\tau}_u | u \in U\}$. To estimate prior parameters $\alpha_1, \beta_1$ for item simplicities we use the same approach as for $\alpha_0, \beta_0$; the only difference is that item simplicities $s_i$ are estimated using error lists $l_i$ computed for every item $i$, rather than for every student $u$.

Table 3 reports the average standard deviation of students and items errors computed over 288 assignment with 25633 items. The grading range is $[0, 100]$.

Table 3 reports the average standard deviation of students and items inferred from the model. As we can see, students are responsible for over two thirds of the overall reviewing error.

### 5.2 Large error behavior

While students intuitively understand that small random errors will be averaged out, they are very concerned by large errors that, they fear, will skew their overall grade. Thus, we are interested in determining whether such large errors are more often due to students who are grossly imprecise, or items that are very hard to grade. In other words: do large errors cluster more around imprecise students, or around hard-to-evaluate items? We can answer this question because in CrowdGrader, items are assigned to students in a completely random way. Thus, any correlation between errors on items or students indicates causality.

We answer this question in two ways. First, we measured the information-theoretic coefficient of constraint. To compute it, let $X$ and $Y$ be two random variables, obtained by sampling uniformly at random two reviews $x$ and $y$ corresponding to the same item, or to the same student, and letting $X$ (resp. $Y$) be 1 if $x$ (resp. $y$) is incorrect by more than a pre-defined threshold (such as, 20% of the grading range for the assignment). Then, the mutual information $I(Y; X)$ indicates the amount of information shared by $X$ and $Y$, and the coefficient of constraint $I(X; Y)/H(X)$, where $H(X)$ is the entropy of $X$, is an information-theoretic measure of the correlation between $X$ and $Y$.

Table 4 gives $I(X; Y)/H(X)$ for student and item errors, for different values of the error choice, and taking as reference truth for each item either the instructor grade, or the Olympic average for the item. When taking instructor grades as reference (Table 4a), large errors are about 5 times more correlated on items than on students, as measured by the coefficient of constraint. When Olympic grades are take as reference (Table 4b), large errors are about as correlated on items as they are on students. The difference in behavior is due to the fact that, when an instructor disagrees with the student-given grades on an item, this generates highly correlated errors on that item with respect to the instructor grade, but not with respect to the Olympic average. In any case, the results show that there is no particular correlation on students.

Another way to measure whether large errors tend to cluster around hard-to-evaluate items or around imprecise students consists in measuring the conditional probability $p_n = P(\xi \geq n | \xi \geq n - 1)$ of an item (resp. student) having $\xi \geq n$ grossly erroneous reviews, given that it has at least $n - 1$. If errors on an item (resp. reviewer) are uncorrelated, we would expect that $p_1 = p_2 = p_3 = \cdots$. If these conditional probabilities grow with $n$, so that $p_2 > p_3 > p_1$, this indicates that the more errors an item (resp. a student) has participated in, the more likely it is that there are additional errors. The values of $p_1, p_2, p_3, \ldots$ allow thus one to form an intuitive appreciation for how clustered around items or students the errors are.

The results are given in Figure 1. The data shows some clustering around users, for large errors of over 30% of the grading range. However, clustering around users seems weaker than clustering around items.

This provides a possible explanation for why reputation systems have not proved effective in dealing with errors in peer-graded assignments with CrowdGrader. Reputation systems are effective in characterizing the precision of each student, and taking it into account when computing each item’s grade. Our results indicate however that errors in CrowdGrader are not strongly correlated with students, limiting the potential of reputation systems.

### 6. STUDENT ABILITY VS. ACCURACY

A natural question is whether better students make better graders. To answer this question, we can approximate the expertise of every student with the grade received by the student’s own submission, and we can then study the correlation between the student’s submission grade, and the review error. As we have only partial coverage of students with instructor grades, we compute the grade received by the student’s own submission via Olympic average, rather than instructor grade. As the two generally are close, this increases coverage with minimal influence on the results. We study grading error with respect to both instructor grades and Olympic average.

<table>
<thead>
<tr>
<th>Error Threshold</th>
<th>10%</th>
<th>15%</th>
<th>20%</th>
<th>25%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students</td>
<td>0.015</td>
<td>0.026</td>
<td>0.017</td>
<td>0.019</td>
<td>0.017</td>
</tr>
<tr>
<td>Items</td>
<td>0.075</td>
<td>0.082</td>
<td>0.082</td>
<td>0.1</td>
<td>0.097</td>
</tr>
</tbody>
</table>

(a) Item errors computed with respect to instructor’s grades. We use only assignments that have at least 30% of items grade by the instructor.

<table>
<thead>
<tr>
<th>Error Threshold</th>
<th>10%</th>
<th>15%</th>
<th>20%</th>
<th>25%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students</td>
<td>0.018</td>
<td>0.018</td>
<td>0.019</td>
<td>0.020</td>
<td>0.021</td>
</tr>
<tr>
<td>Items</td>
<td>0.045</td>
<td>0.030</td>
<td>0.020</td>
<td>0.021</td>
<td>0.020</td>
</tr>
</tbody>
</table>

(b) Item errors computed with respect to Olympic average.

Table 4: Coefficient of constraint $I(X, Y)/H(X)$ of large errors on the same item or by the same student, for different error thresholds.
First, for each assignment independently, we sort all students according to their \( x \)-value, and we assign them to one of 10 percentile bins: if the assignment comprises \( n \) students and the student ranks \( k \)-th, the student will be in the \( \lceil 10k/n \rceil \) bin; we call these bins the 10\%, 20\%, \ldots, 100\% bins. For each assignment \( a \), we normalize the grading range to [0, 100], and we let \( n_{a,q} \) and \( e_{a,q} \) be the number of students and the average error in the \( q \) percentile bin of assignment \( a \), respectively. The average error for assignment \( a \) overall is thus \( e_a = \frac{\sum n_{a,q} e_{a,q}}{\sum n_{a,q}} \). There are two ways of measuring the average error \( e_{a,q} \) for one bin: as average absolute value error, or as average root-mean-square error. The two approaches lead to qualitatively similar conclusions, as we show later in this section. Due to lack of space, unless otherwise explicitly stated, we present here only the results for average absolute value, as they are somewhat less sensitive to rare large errors, and thus, more stable. The complete set of results is reported in [9].

We aggregate data from multiple assignments, computing for each percentile bin an absolute and a relative error, as follows. The absolute error \( e_q \) for each percentile \( q \) is computed as

\[
e_q = \sum_{a} n_{a,q} e_{a,q} / \sum_{a} n_{a,q}.
\]

The relative error \( r_q \) for each percentile \( q \) is computed as

\[
r_q = \sum_{a} n_{a,q} (e_{a,q}/e_a) / \sum_{a} n_{a,q}.
\]

where \( e_{a,q}/e_a \) is the relative error of bin \( q \) in assignment \( a \).

### 6.2 Student ability vs. error

The data reported in Figure 2b shows the existence of some correlation between student submission grade, and grading precision, measured with respect to the Olympic average. In relative terms, students in the 80–100\% percentile brackets show error that is 10\% to 20\% greater than students with higher submission grade. The absolute error tells a similar story. The two graphs do not have the same shape, due to the fact that relative errors are computed in (2) in a per-assignment fashion. In Figure 2a we report the same data, computed using rms error rather than average absolute value error. The data is qualitatively similar. Due to lack of space, in the remaining graphs we consider only average absolute error.

In Figure 3 we compare the error with respect to Olympic average with the error compared to instructor grades, for the subset of classes where at least 30\% of submissions have been instructor-graded. While the absolute values are different, we see that the curves are very closely related, indicating that Olympic averages are a good proxy for instructor grades when studying relative changes in precision. The error with respect to instructor grades has very wide error bars for the 90\% percentile, mainly due to the low number of data points we have for that percentile bracket in our dataset. We favor the comparison with the Olympic average, since the abundance of data makes the statistics more reliable.

The correlation between student ability (as measured by the submission score) and grading precision is lower than we expected. This might be a testament to the clarity of the rubrics and grading instructions provided by the instructors: apparently, such instructions ensure that most students are able to grade with reasonable precision the work by others. This may also be a consequence of the fundamental skill and background homogeneity of students in a classroom, as compared to a MOOC. We note that [2] also reported...
and instructor grades confirms that the Olympic average is a good proxy for studying variation with respect to instructor grade also. We omit the analogous of Figure 3 for the timing analysis due to lack of space; similarly, we include results only for mean absolute error. The complete result set is available in [9].

Time to complete a review. We first considered the correlation between the time spent by students performing each review, and the accuracy of the review; the results are reported in Figure 4. The results indicate that reviews that are performed moderately quickly tend to be slightly more precise. The correlation is weaker than we expected. We expected to find error peaks due to students that spent very little time reviewing, and that entered a quick guess for the submission grade, rather than performing a proper review. There are no such peaks: either students are very good at quickly estimating submission quality, or they mostly take reviewing and seriously in CrowdGrader. We believe the latter hypothesis is likely the correct one: for instance, in many computer science assignments, there is no good way of “eye-balling” the quality of a submission without compiling and running it.

Time at which a review is completed. Next, we studied the correlation between the absolute time when reviews are performed, and the precision of the reviews. Figure 5 shows the existence of a modest correlation: the reviews that are completed in the first 10% percentile tend to be 10% more accurate than later reviews. The effect is rather small, however. In a typical CrowdGrader assignment, students are given ample time to complete their reviews, and the reviews themselves take only one hour or so to complete. Students likely do not feel they are under strong time pressure to complete the reviews, and time to deadline has little effect on accuracy.

Order in which reviews are completed. Lastly, we study whether the order in which a student performs the reviews affects the accuracy of the reviews. We are interested in the question of whether students learn while doing reviews, and become more precise, or whether they grow tired and impatient as they perform the reviews, and their accuracy decreases. Figure 6 shows that the accuracy of
students does not vary significantly as the students progress in their review work. Evidently, the typical review load is sufficiently light that students do not suffer from decreased attention while completing the reviews.

Figure 4: Absolute and relative grading error vs. the time employed to perform a review; the first percentile bin 10% corresponds to reviews with shortest review time. The grading range is normalized to [0, 100], and the error is measured with respect to the Olympic average. The error bars indicate one standard deviation.

Figure 5: Absolute and relative grading error vs. absolute time when a review is completed. The first percentile bin 10% corresponds to the 10% of reviews that were completed first among all assignment reviews. The grading range is normalized to [0, 100], and the error is measured with respect to the Olympic average. The error bars indicate one standard deviation.

8. TIT-FOR-TAT IN REVIEW FEEDBACK

In CrowdGrader, students can leave feedback to each review and grade they receive. The feedback is expressed via 1-to-5 star rating systems as follows:

- 1 star: factually wrong; bogus.
- 2 stars: unhelpful.
- 3 stars: neutral.
- 4 stars: somewhat helpful.
- 5 stars: very helpful.

Many such ratings are given as tit-for-tat: when a student receives a low grade, the student responds by assigning a low feedback score (typically, 1 star) to the corresponding review. Indeed, CrowdGrader includes a technique for identifying such tit-for-tat, so that students, whose overall grade depends also on the helpfulness of their reviews, are not unduly penalized. We were interested in analyzing the question of how prevalent tit-for-tat is.

Overall, review grade and review feedback have a correlation of 0.39, with a p-value smaller than $10^{-300}$. The correlation between grade and feedback indicates tit-for-tat, as there is no reason why lower grades should per-se be associated with written reviews that are less helpful. Interestingly, the correlation is fairly independent from the subject area. To bring the tit-for-tat into sharper evidence, we computed also the following statistics. We consider a grade a $p$ (resp. $n$) outlier if the grade is over 20% above (resp. below) the Olympic average. We then measured the conditional probabilities $P_p$, $P_n$, that $p$ and $n$ outliers would receive a one or two-star rating, conditioned over the probability that the reviews received a rating at all (students do not always rate the reviews they receive). Over all assignments, we measured $P_p = 0.06$ and $P_n = 0.44$. Since there is no a-priori reason why overly negative reviews may be of worse quality than overly positive ones, the excess probability $P_n = P_p = 0.38$ can be explained by tit-for-tat. This shows that tit-for-tat is rather common: for grades that are 20% or more below the consensus, there is a 38% probability of low feedback due to tit for tat. Fortunately, it is easy to discard low ratings given in response to below-average grades, as CrowdGrader does.

9. DISCUSSION

We presented an analysis of a large body of peer-grading data, gathered on assignments that used CrowdGrader across a wide set of subjects, from engineering to business and humanities. Our main interest consisted in identifying the factors that influence grading errors, so that we could devise methods to control or compensate for such factors. Our results can be thus summarized:

- Large errors are no more strongly correlated on students than
they are on items. In other words, students who are imprecise on many submissions are not a dominant source of error.

- There is some correlation between the quality of a student’s own submission (which is an indication of the student’s accomplishment), and the grading accuracy of the student, but the correlation is weak and limited to the student with highest, and lowest submission grades.
- There is little correlation between the accuracy of a review, and the time it took to perform the review, or how late in the review period the review was performed.
- There is clear evidence of tit-for-tat behavior when students give feedback on the reviews they receive.

All of the correlations we measured, except for the tit-for-tat one, are rather weak. This is a reassuring confirmation that peer-grading works as intended. There are no large sources of uncontrolled error due to factors such as student fatigue in doing the reviews, or gross inability of weaker students to perform the reviews. The peer-grading tool, in our classroom settings, ensures that the remaining errors are fairly randomly distributed, with little remaining structure.

The results highlight the difficulties in using reputation systems to compute submission grades in peer-grading assignments in high-school and university settings. Reputation systems characterize the behavior of each student, in terms for instance of their grading accuracy and bias, and compensate for each student’s behavior when aggregating the individual review grades into a consensus grade. However, our results indicate that the large errors that most affect the fairness perception of peer grading are most closely associated with items, rather than with students. Reputation systems are powerless with respect to errors caused by hard-to-grade items: even if they can correctly pinpoint which submissions are hard to grade, little can be done except flagging them for instructor grading. Indeed, the reputation system approach of [14], which yielded error reductions of about 30% for MOOCs, yielded virtually no benefit in our classroom settings.

There is more potential, instead, in approaches that make it easier to grade difficult submissions. In CrowdGrader, we introduced anonymous forums, associated with each submission, where submissions authors and reviewers can discuss any issues that arise while viewing the submission. These forums are routinely used, for instance, to solve the glitches that often arise when trying to compile or run code written by someone else. Anecdotally, these forums have markedly increased the satisfaction with the peer-grading tool, as students feel that they have a safety net if they make small mistakes in formatting or submitting their work, and are in the loop should any issues occur.

10. ACKNOWLEDGEMENTS
This research has been supported in part by the NSF Award 1432690.

11. REFERENCES
Sequence Matters, But How Exactly?
A Method for Evaluating Activity Sequences from Data

Shayan Doroudi\textsuperscript{1}, Kenneth Holstein\textsuperscript{2}, Vincent Aleven\textsuperscript{2}, Emma Brunskill\textsuperscript{1}

\textsuperscript{1}Computer Science Department, \textsuperscript{2}Human-Computer Interaction Institute
Carnegie Mellon University
{shayand, kjholste, aleven, ebrun}@cs.cmu.edu

ABSTRACT
How should a wide variety of educational activities be sequenced to maximize student learning? Although some experimental studies have addressed this question, educational data mining methods may be able to evaluate a wider range of possibilities and better handle many simultaneous sequencing constraints. We introduce Sequencing Constraint Violation Analysis (SCOVA): a general method for evaluating alternative activity sequences using existing data. SCOVA can be used to explore many complex sequencing constraints, such as prerequisite relationships, blocking, interleaving, and spiraling. We demonstrate SCOVA on data collected from a fractions intelligent tutoring system (ITS). Some of our findings challenge our initial hypotheses regarding sequencing, illustrating the utility and versatility of the method. The method can also be applied to other learning environments, as long as the available data has substantial variability in students' activity sequences.

1. INTRODUCTION
How does the sequencing of pedagogical activities impact student learning? Answers to this question can both contribute to core learning sciences knowledge, as well as have important practical implications for how educational activities should be sequenced in order to maximize learning. As such, there has been significant interest in this issue, and prior research suggests that student learning can be quite sensitive to temporal sequencing (e.g., [16, 1, 15, 17]).

Prior work that tackles this problem mainly falls into either theoretical analyses or empirical studies. Unfortunately, conducting theoretical analyses of the cognitive demands of individual tasks and the interdependencies among multiple tasks [7, 10, 3] can be prohibitively time-consuming for large curricula. In addition, such analyses may be particularly vulnerable to various cognitive biases, such as expert blindness [12]. Considerable experimental research has examined the effects of activity sequencing along various dimensions, including interleaving versus blocking of topics [1, 17] and sequencing of activities according to the degree of scaffolding they provide [15, 8]. However, such classroom experimental studies typically compare only two or three possible conditions, in contrast to the enormous number of orderings possible (at least exponential in the number of activity categories of interest).

An educational data mining approach could allow us to evaluate a much broader range of possible orderings in order to better understand which sequences may be optimal. Moreover, it might be possible to apply such techniques to any datasets that have considerable variation in how they order instructional content for students. These include datasets generated from educational technologies that present activities in a partially or fully randomized order (e.g., [13]), those that adaptively present activities in response to measured student variables (e.g., [4]), and those that provide students with some degree of control over activity selection (e.g., [11]).

We are particularly interested in investigating which orderings over a variety of topics and activity types are most effective for maximizing student learning and performance. Prior educational data mining approaches have focused on examining pairwise dependencies between instructional items (e.g., individual skills, problems, or problem sets) in a curriculum, in order to infer underlying prerequisite structure [5, 21, 18]. The prerequisite structures learned via such methods could be used, for example, to inform adaptive problem selection algorithms that avoid presenting a given item until the student is believed to have mastered its prerequisites [7]. Other methods for detecting ordering effects over instructional items have additionally relied upon the use of fitted Bayesian Knowledge Tracing (BKT) models [13, 19], and have thus depended upon strong assumptions about student learning. Whereas these prior approaches are typically limited to discovering pairwise relationships between items, and have tended to assume that these items are presented in a blocked fashion, we wish to examine the impacts on student learning and performance of more complex (and potentially softer) sequencing constraints.

We investigate the question of optimal topic and activity type sequencing in the context of our fractions intelligent tutoring system (ITS) [6]. Our tutor covers three broad topics (making and naming fractions, fraction equivalence and ordering, and fraction addition) and three different types of activities that correspond to learning mech-
anisms in the theoretical Knowledge-Learning-Instruction (KLI) framework: sense-making, induction and refinement, and fluency-building processes [9]. While previous experimental work has investigated the optimal sequencing of activity types under the KLI framework [14], there has been little empirical work investigating the optimal sequencing of topics in a fractions curriculum, and no work to our knowledge examining how the optimal sequencing of activity types may vary across topics.

We develop a general-purpose method for leveraging log data to evaluate and compare different ways of sequencing activities. We believe our method for evaluating sequencing constraints can be utilized to discover how to sequence activities in a variety of learning environments. We tested our method on log data from our fractions tutor and found results that countered our initial hypotheses on how to order both topics and activity types. We also found that the optimal ordering over KLI activity types may vary from topic to topic, but that for the most part, these orderings were consistent with what was suggested by prior literature [14].

2. SEQUENCING CONSTRAINT VIOLATION ANALYSIS (SCOVA)

We first describe our general method, and then present the particular instantiations of our method that we used in our analyses in Section 3. Sequencing Constraint Violation Analysis (SCOVA) is a method for analyzing different sequencing constraints and identifying which ones lead to the best student performance. SCOVA takes as input a set of student trajectories (which contains the sequence of problems given to each student and the students’ responses to those problems) and a cost function for each set of sequencing constraints that one wants to evaluate. The cost function is a function over student trajectories that specifies how often a particular set of sequencing constraints is violated; in particular, it assigns to each student’s sequence a number of violations up to the total length of the sequence.

Many different types of sequencing constraints can be considered. For example, one sequencing constraint could be that a student must be given at least one instance of problem type $X$ before the student is given problem type $Y$. For this constraint, whenever problem $Y$ is presented to a student before any instance of problem $X$, that student trajectory incurs one violation. Another constraint could be that problem $X$ should always appear immediately before problem $Y$, so whenever a student sees problem $Y$ without seeing problem $X$ right before it, that sequence incurs a violation. For such constraints, the cost function is simply the number of problems where the constraint is violated. However, another sequencing constraint could suggest that a student’s trajectory should match a particular desirable sequence, and our cost function in that case could be the Levenshtein distance\(^1\) between the student’s sequence and the desirable sequence. We can also consider sets of more than one sequencing constraints: for example, the constraints could specify that problem $X$ should come before problem $Y$ and problem $Y$ should come before problem $X$. In this case, the cost function counts every time any constraint is violated.

Unlike many existing methods (e.g., [13, 21, 19]), SCOVA is not limited to evaluating pairwise orderings. Indeed, SCOVA can handle much more general constraints on order sequencing, such as blocking, interleaving, and spiraling. SCOVA can also handle constraints that depend not just on the prior history of problems given, but also on the student’s performance and interactions (such as performance on prior activities, pretest score, or measures of affect).

Given the cost functions and student trajectories, SCOVA proceeds as follows for each set of sequencing constraints that we want to evaluate. We first use the cost function to compute the proportion of violations for every student’s sequence by dividing the cost of the sequence by the length of the sequence. We next use the proportion of violations as an input variable in a linear regression model that predicts some measure of student performance (e.g., within-tutor performance, posttest score, or learning gains), and fit the parameters that maximize the log likelihood of the resulting model.

To evaluate the impact of a particular set of sequencing constraints, we look at two measures. First, we compute the Bayesian Information Criterion (BIC) of the linear regression model fit for violations of those constraints. This provides us with a way to compare different sequencing constraints; a model with a lower BIC score provides a better fit of the student data (as evaluated by log likelihood, adjusted for the number of parameters of the model). However, BIC alone simply measures predictive fit, not whether the sequencing constraints are beneficial for students or harmful. To understand whether the sequencing constraints may have a positive or negative impact on the outcome variable, we look at the sign of the coefficient of the violation variable in the fit linear model. We limit our attention to models where the proportion of violations has a negative coefficient—that is, models where violating the sequencing constraints is associated with worse student performance. Among these models, we can then compare the sequencing constraints by comparing the BICs of their models.

Recall that SCOVA can handle multiple sequencing constraints conjunctively (e.g., example problem $X$ should come before $Y$ and $Y$ before $Z$). This makes the most sense when the different sequencing constraints are mutually exclusive, i.e., we cannot incur more than one violation on any particular problem. However, we may want to consider different sequencing constraints that can occur simultaneously and perhaps constrain different aspects of student trajectories (e.g., for example one might constrain the ordering of topics and the other might constrain the ordering of activity types). SCOVA can be extended to simultaneously consider the impact of these different sequencing constraints disjunctively. To do so, we learn a predictive linear regression model with one input variable for each set of sequencing constraints. When we have more than one set of sequencing constraints in our model, we focus our attention on models that have negative coefficients for every predictor corresponding to violations of sequencing constraints. If the BIC of a model
that takes two sequencing constraints into account is lower than that of each of the models that consider just one of the sequencing constraints individually, it suggests that both ordering constraints are important but capture different aspects of student performance. We can also compare the relative effects of violating different sequencing constraints by comparing the coefficients within the same model.

3. EVALUATION DOMAIN

As a concrete example, we now describe how we used our proposed approach to evaluate the impact of ordering on student learning and performance when using an online fractions tutor for fourth and fifth grade fractions topics [6]. The tutor covers topics emphasized in the Common Core, a set of non-binding national standards for mathematics education in the US: naming and fractions on the number line (MN), fraction equivalence and ordering (EQ), and fraction addition (ADD). The tutor was originally developed to investigate the potential benefits of using a broader range of instructional activity types than is typical of an ITS. Tutor activities were designed to promote each of the 3 categories of learning mechanisms posited under the KLI framework [9]: sense-making (SM), induction and refinement (IR), and fluency-building (F). The tutor’s curriculum includes activities targeting each of these categories of learning mechanisms, for each of the main topics.

Under KLI, SM processes correspond to “explicit, verbally mediated learning in which students attempt to understand or reason” [9]. IR processes are defined as non-verbal learning processes that improve the accuracy of knowledge, and fluency processes are non-verbal processes that strengthen memory and enable students to apply their procedural knowledge faster and more fluently. As such, SM activities in our tutor were designed to promote conceptual understanding through an interleaving of brief instructional videos with exercises intended to support self-explanation. By contrast, IR activities in our fractions tutor were designed to emphasize procedural learning and practice via fine-grained task decomposition and step-level guidance – as is typical of ITSs [20]. Finally, fluency-building activities were designed to promote the development of fluent performance on minimally decomposed problem-solving exercises. A more detailed description of our operationalization of these three activity types can be found in [6].

3.1 Sequencing Constraints

We consider a variety of sequencing constraints over both topics and activity types in our analyses. Since we have three topics and three activity types there are six potential orderings of each. For each of the following constraints (aside from the baselines at the end) we consider them with respect to each of the six possible orderings (for either topics or activity types).

3.1.1 Exposure-Based Constraints

Exposure-based constraints stipulate that students be exposed to (i.e., carry out) one topic/activity type a certain number of times before being exposed to the next. Every time the student receives a problem before being exposed to its “prerequisite” enough times, a violation is incurred. We define two categories: Exposure-based topic constraints require that students do at least one problem of a topic before seeing a problem of the next topic. Exposure-based type constraints require that within each topic, students should do one problem of an activity type before seeing the next activity type, without constraining the order of topics. Note that we can have the ordering over activity types fixed for every topic, or we can let it vary. If we let it vary, there are $6^2 = 216$ possible exposure-based varying type constraints.

3.1.2 Performance-Based Constraints

Performance-based constraints stipulate that students should reach a certain level of within-tutor performance on a topic/activity type before being exposed to the next. Every time the student receives a problem when their recent performance on its “prerequisite” is not beyond some threshold, a violation is incurred. Notice that even though such a constraint may be satisfied for a given student at a certain point in time, it is possible that it will no longer be satisfied later on, if the student’s performance drops. Performance-based topic constraints require that students’ performance on the last 10 steps of the topic should be beyond some topic-specific threshold before they receive problems for the next topic. (These steps may be from one problem or span over several problems.) By contrast, performance-based type constraints require that within each topic, students’ performance on the last 10 steps on a particular activity type should be beyond some threshold specific to that topic-type pair before they receive problems of the next activity type (for the given topic). As before, in addition to the six type constraints that are fixed per topic, we have 216 possible performance-based varying type constraints.

We selected thresholds to detect a basic level of competency with problems of a particular activity type within a topic—a lower bar than mastery. The thresholds shown in Table 1 were obtained by taking the average student performance on the last 10 steps upon doing two problems of the given topic or topic-type pair.

3.1.3 Blocking and Interleaving-N Constraints

To show the flexibility of the SOCOVA method in considering sequencing constraints beyond straightforward prerequisite relationships, we consider whether topics and activity types should be interleaved or blocked with respect to topics/types. We measure violations in terms of Levenshtein distance from a particular sequence. The blocking topic constraint stipulates that for every student, the first third of their sequence (rounding up) should correspond to the first topic, the second third (rounding up) should correspond to the second topic, and the last third should correspond to the last topic. This is not a sequence we would typically be able to assign in practice, because we do not generally know how many problems a student will do ahead of time, but it represents a pure form of blocking while guaranteeing students see all of the activity types.

The interleaving-N topic constraints, for $N = 1, \ldots, 6$, require sequences that

---

2In the fractions tutor, activities within each of these three broad topics broke down further into multiple subtopics. For example, fraction equivalence and ordering included activities on finding common denominators, reducing fractions, and identifying equivalent fractions using number lines, among other subtopics. In addition, individual activities typically targeted a number of finer-grained skills.
Table 1: Thresholds used for performance-based topic and type constraints. Notice that for the type constraints, we have distinct thresholds for each topic. The thresholds were obtained by taking the average student performance on the last 10 steps upon doing two problems of the given topic or topic-type pair.

<table>
<thead>
<tr>
<th>Topic/Type</th>
<th>Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>MN</td>
<td>0.453</td>
</tr>
<tr>
<td>EQ</td>
<td>0.360</td>
</tr>
<tr>
<td>ADD</td>
<td>0.206</td>
</tr>
<tr>
<td>MN/SM</td>
<td>0.415</td>
</tr>
<tr>
<td>MN/IR</td>
<td>0.514</td>
</tr>
<tr>
<td>MN/F</td>
<td>0.125</td>
</tr>
<tr>
<td>EQ/SM</td>
<td>0.356</td>
</tr>
<tr>
<td>EQ/IR</td>
<td>0.547</td>
</tr>
<tr>
<td>EQ/F</td>
<td>0.308</td>
</tr>
<tr>
<td>ADD/SM</td>
<td>0.262</td>
</tr>
<tr>
<td>ADD/IR</td>
<td>0.158</td>
</tr>
<tr>
<td>ADD/F</td>
<td>0.269</td>
</tr>
</tbody>
</table>

give $N$ problems of the first topic followed by $N$ problems of the second topic followed by $N$ problems of the third topic. However, if a student did less than 3$N$ problems in total, we instead use the sequence used for the blocking constraint, in order to check whether they get reasonable exposure to all three topics.

### 3.1.4 Proportion-Only Baselines
To see if ordering topics or activity types actually matters, we compare to baselines that just use the proportions of topics or activity types in the sequence as predictors to predict within-tutor performance. Note that our two baselines each have two predictors (e.g., for activity types, we have one for proportion of SM and proportion of IR; the proportion of fluency-building activities is linearly dependent on the first two and so it is not needed in the model).

### 3.2 Hypotheses
We started data analysis with several hypotheses about the best order of topics and activity types. We note however that in order to illustrate our method, the specific hypothesized best order does not matter, although it does matter in illustrating that the method can produce unexpected (but reasonable) results.

#### 3.2.1 Topic Dependencies
Our first hypothesis is that in early fractions learning, topics build on each other in the following way. MN helps students build a basic representation of fractions as numbers that have a magnitude, represented by their place on the number line. This representation is hypothesized to help in building an understanding of the notion of equivalence and the notion that fractions can be compared and ordered in terms of their magnitude. Moreover, equivalence would appear to be a strict prerequisite for addition of fractions with unlike denominators, because fractions with unlike denominators need to be converted to equivalent fractions before they can be added. Thus, the hypothesized best topic order is MN-EQ-ADD. Topics may not need to be fully blocked (i.e., presenting all MN activities before any EQ activities, and all EQ activities before any ADD activities), but it may be better for students to initially be exposed to topics in this order and perhaps continue to see the different topics in an interleaved fashion (as interleaving has been show to be beneficial [1, 17]).

#### 3.2.2 Type Dependencies
As mentioned, the KLI framework distinguishes between three distinct classes of learning mechanisms, SM, IR, and F. It does not, however, make any claims regarding the order in which these processes might be most effective or even whether each class of mechanisms is needed when learning in a complex domain (such as fractions). There has been little prior work investigating how instructional activities targeting each of the KLI activity types can best be sequenced to maximize student learning and performance. However, [14] previously found that presenting students with SM activities before presenting them with fluency-building activities is beneficial when teaching connection making between multiple graphical representations of fractions. Given the dearth of prior work in this area, we do not have very strong expectations regarding the best order of these different activity types within a topic. However, in line with the work by [14], our hypothesis is that SM-targeting activities should come first, then IR-targeting activities, and finally, F-targeting activities. A second reason to expect that it is effective to do IR activities before F activities is that in our tutors, IR activities provide more elaborate scaffolding than F activities. As before, we do not mean to suggest a fully blocked ordering may be best, but also consider orders that interleave activity types with the hypothesized SM-IR-F order strictly observed early on.

### 3.3 Data
We collected data from 347 students using our ITS (in 20 classrooms across four different schools). The data was initially collected for a randomized control trial comparing three adaptive problem selection policies and two non-adaptive policies. The three adaptive policies had quite a bit of variation in the kinds of trajectories given to students; they thus provide data that is a good fit for SCOVA. However, the non-adaptive policies resulted in trajectories that were identical in how they sequenced topics and activity types, so we did not use data from those policies in our analyses (leaving 211 students). Students were given a pretest, followed by using the tutor for typically four class periods, and were finally given a posttest that was identical to the pretest. Each student worked at their own pace and completed as many problems as they could during the allotted time, resulting in a tail of students who did many more problems than average. This could present a confound in our analysis since students who do many problems are more likely to be high performing students, as well as violating sequencing constraints less than others (because they are likely to do many problems after satisfying all sequencing constraints). We thus limited our analyses to students who did 60 or fewer problems (197 students).

### 3.4 Modeling
In the SCOVA framework, we fit a linear regression model with predictors corresponding to the proportion of violations of one or more sets of sequencing constraints. The outcome variable we used was the within-tutor performance of students on all problems of the tutor with each topic-type pair having an equal weight (e.g., each student’s performance on MN/SM problems has an equal weight to their performance on EQ/F problems). If a student received no problems of a
topic-type pair, then the average is only over the topic-type pairs they received. One could also add other predictors to improve the model fits and potentially control for other confounds. We add the student’s pretest score as a predictor to all of our models as this improved the model fit.

4. RESULTS

Table 2 shows the BICs of models with only a single ordering constraint predictor corresponding to performance-based and exposure-based topic and type sequencing constraints in addition to BICs of the two proportion-based baselines. First, we notice that the lowest BIC models using exposure-based and performance-based ordering constraints have a better fit than the baseline models, which, as mentioned, only consider the proportion of activities given for either topic or activity type. This suggests that ordering of topics and activity types makes a difference beyond just the frequency with which they appear.

Table 2: Comparison of BICs of individual exposure-based and performance-based constraints as well as proportion-only baselines. Aside from the proportion-only baselines, BICs corresponding to models where the coefficient of the predictor is negative are shown in bold. The smallest BIC in each column is underlined.

<table>
<thead>
<tr>
<th>Topic Constraints</th>
<th>Exposure</th>
<th>Performance</th>
<th>Type Constraints</th>
<th>Exposure</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>MN-EQ-ADD</td>
<td>-236.28</td>
<td>-299.69</td>
<td>SM-IR-F</td>
<td>-226.16</td>
<td>-226.84</td>
</tr>
<tr>
<td>EQ-MN-ADD</td>
<td>-244.39</td>
<td>-319.13</td>
<td>IR-SM-F</td>
<td>-208.59</td>
<td>-218.94</td>
</tr>
<tr>
<td>MN-ADD-EQ</td>
<td>-201.04</td>
<td>-274.17</td>
<td>SM-F-IR</td>
<td>-193.39</td>
<td>-200.89</td>
</tr>
<tr>
<td>EQ-ADD-MN</td>
<td>-201.26</td>
<td>-254.75</td>
<td>IR-F-SM</td>
<td>-196.85</td>
<td>-217.20</td>
</tr>
<tr>
<td>ADD-MN-EQ</td>
<td>-193.81</td>
<td>-199.80</td>
<td>F-SM-IR</td>
<td>-202.91</td>
<td>-224.57</td>
</tr>
<tr>
<td>ADD-EQ-MN</td>
<td>-205.73</td>
<td>-193.84</td>
<td>F-IR-SM</td>
<td>-192.97</td>
<td>-200.32</td>
</tr>
<tr>
<td>Proportion-Only</td>
<td>-233.48</td>
<td></td>
<td>Proportion-Only</td>
<td>-201.77</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Comparison of BICs of models combining exposure-based topic and type constraints. BICs corresponding to models where the coefficients of both predictors are negative are shown in bold. The smallest BIC is underlined.

<table>
<thead>
<tr>
<th>Type Constraints</th>
<th>Exposure</th>
<th>Performance</th>
<th>Type Constraints</th>
<th>Exposure</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM-IR-F</td>
<td>-246.09</td>
<td>-232.81</td>
<td>IR-SM-F</td>
<td>-232.95</td>
<td>-231.28</td>
</tr>
<tr>
<td>SM-F-IR</td>
<td>-247.24</td>
<td>-240.24</td>
<td>IR-F-SM</td>
<td>-202.08</td>
<td>-203.99</td>
</tr>
<tr>
<td>F-SM-IR</td>
<td>-205.57</td>
<td>-200.89</td>
<td>F-SM-IR</td>
<td>-200.48</td>
<td>-208.98</td>
</tr>
<tr>
<td>F-IR-SM</td>
<td>-217.35</td>
<td>-201.94</td>
<td>F-IR-SM</td>
<td>-210.61</td>
<td>-201.07</td>
</tr>
</tbody>
</table>

We also find that the models that put fractions addition first either have the worst BICs or have positive coefficients (i.e., violation of constraints correlates with increased student performance), which makes sense, as we really do not think students should be doing addition (potentially with unlike denominators) before fraction equivalence. Likewise, the models with the best BICs and largest negative coefficients are the ones that put ADD last.

Finally, we find that the performance-based constraints have lower BICs than the exposure-based constraints. This reasonably seems to suggest that students’ within-tutor performance can be predicted more accurately when we take into account the extent to which individual students reached a basic level of competence on one topic/type before being exposed to the next topic/type. We must note, however, that for the performance-based metric, the number of violations is impacted by a student’s performance, and is thus related to the outcome variable in a confounded way. For example, a student who does very well on the tutor would be more likely to get fewer performance-based violations for any sequence than a student who does poorly on the tutor, partially explaining the lower BICs for performance-based models than exposure-based models. While we cannot conclude that performance-based constraints are better than exposure-based constraints from this analysis, we hypothesize that the relative ranking of different orders of topics/types may not be impacted severely by this confound.

To start to understand the interaction of type and topic ordering constraints on within-tutor student performance, we fit linear regression models that used two prerequisite violation input variables: one for one of the six topic orderings, and one for one of the six type orderings. Table 3 shows
The BICs for all 36 models that have pairs of violations of exposure-based topic and type constraints as predictors, and Table 4 shows analogous results for pairs of performance-based constraints. We find that both for exposure-based and performance-based constraints, the model with the lowest BIC uses the EQ-MN-ADD ordering over topics, but for exposure-based constraints the ordering over activity types is IR-SM-F, while for performance-based constraints it is IR-F-SM. Note that this is different from the lowest BIC ordering of activity types when using only type constraints (SM-IR-F, see Table 2). However, we find that for many other orderings over topics (e.g., MN-EQ-ADD and MN-ADD-EQ), the model with the lowest BIC is the one with the SM-IR-F ordering over activity types. This suggests that the best ordering over activity types may depend on how we sequence the topics.

Indeed, the best ordering over activity types might vary from topic to topic (e.g., to maximize student performance it may be best to give IR first for EQ but SM first for MN). To test this possibility, we searched for the lowest BIC model with a predictor corresponding to some varying type constraints and a predictor for one of the six topic constraints. The lowest BIC model according to exposure-based constraints suggests the ordering IR-SM-F for EQ, SM-IR-F for MN, and F-IR-SM for ADD, and the lowest BIC model according to performance-based constraints suggests the ordering IR-SM-F for EQ, SM-IR-F for MN, and IR-SM-F for ADD. Table 5 shows the coefficients and fits for both of these lowest BIC models. Notice that the coefficients for the topic constraints have larger magnitudes than those for the varying type constraints (although not much larger in the exposure-based model), suggesting that sequencing over topics is more important than sequencing over activity types. Moreover, the coefficients of the topic and activity type constraints violation variables in Table 5 are not only highly significant (i.e., significantly different from 0), but also their magnitudes are quite substantial given the outcome variable is bounded between 0 and 1. This suggests that students who receive activities in an order that has a large proportion of sequencing constraint violations would be expected to have considerably worse performance on the tutor problems.

Finally, we turn to models based on blocking and interleaving constraints. Table 6 shows the results comparing interleaving-\(N\) constraints and blocking constraints for all six orderings over topics. Again we find that the model corresponding to the EQ-MN-ADD order has the lowest BIC, but interleaved in chunks of four problems. This agrees with our hypothesis that one should not simply present the topics in a blocked fashion. Interestingly, most of the other models, including ones corresponding to fully interleaving or blocking, have equally bad BICs, regardless of the topic order.

Table 4: Comparison of BICs of models combining performance-based topic and type constraints. BICs corresponding to models where the coefficients of both predictors are negative are shown in bold. The smallest BIC is underlined.

<table>
<thead>
<tr>
<th>Model</th>
<th>SM-IR-F</th>
<th>IR-SM-F</th>
<th>SM-F-IR</th>
<th>IR-F-SM</th>
<th>F-SM-IR</th>
<th>F-IR-SM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MN-EQ-ADD</td>
<td>-319.39</td>
<td>-297.22</td>
<td>-301.80</td>
<td>-298.25</td>
<td>-299.90</td>
<td>-296.39</td>
</tr>
<tr>
<td>EQ-MN-ADD</td>
<td>-328.84</td>
<td>-330.35</td>
<td>-314.33</td>
<td>-336.70</td>
<td>-330.46</td>
<td>-317.38</td>
</tr>
<tr>
<td>MN-ADD-EQ</td>
<td>-300.02</td>
<td>-285.10</td>
<td>-270.36</td>
<td>-283.20</td>
<td>-286.98</td>
<td>-269.96</td>
</tr>
<tr>
<td>EQ-ADD-MN</td>
<td>-269.67</td>
<td>-280.47</td>
<td>-249.80</td>
<td>-279.55</td>
<td>-261.14</td>
<td>-250.23</td>
</tr>
<tr>
<td>ADD-MN-EQ</td>
<td>-239.09</td>
<td>-215.61</td>
<td>-203.15</td>
<td>-214.25</td>
<td>-220.07</td>
<td>-199.02</td>
</tr>
<tr>
<td>ADD-EQ-MN</td>
<td>-233.34</td>
<td>-213.69</td>
<td>-196.73</td>
<td>-211.92</td>
<td>-219.29</td>
<td>-195.28</td>
</tr>
</tbody>
</table>

Table 5: Best fitting models incorporating both topic constraints and varying type constraints. The lowest BIC model according to exposure-based constraints suggests IR-SM-F for EQ, SM-IR-F for MN, and F-IR-SM for ADD, and the lowest BIC model according to performance-based constraints suggests the ordering IR-SM-F for EQ, SM-IR-F for MN, and IR-SM-F for ADD.

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>p-value</th>
<th>Coefficient</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>0.37</td>
<td>0.45</td>
<td>&lt; 2 * 10^-16</td>
</tr>
<tr>
<td>Pretest</td>
<td>0.025</td>
<td>0.023</td>
<td>4.77 * 10^-10</td>
</tr>
<tr>
<td>Topic Violations</td>
<td>-0.20</td>
<td>8.07 * 10^-7</td>
<td>-0.36</td>
</tr>
<tr>
<td>Type Violations</td>
<td>-0.17</td>
<td>4.20 * 10^-6</td>
<td>-0.22</td>
</tr>
<tr>
<td>BIC</td>
<td>-260.77</td>
<td>-355.00</td>
<td></td>
</tr>
<tr>
<td>Adjusted r^2</td>
<td>0.39</td>
<td>0.62</td>
<td></td>
</tr>
</tbody>
</table>

5. DISCUSSION

Our novel method for evaluating activity sequences led to a number of interesting findings about sequencing topics and activity types in our tutor, illustrating the utility of the method. We found that all of the models fit using various topic sequencing constraints unanimously suggested that...
Interleaving-1  Interleaving-2  Interleaving-3  Interleaving-4  Interleaving-5  Interleaving-6  Blocking
MN-EQ-ADD -193.09  -201.03  -198.85  -198.33  -197.80  -195.52  -195.63
EQ-MN-ADD -195.89  -197.01  -198.89  -211.94  -202.93  -194.93  -193.91
MN-ADD-EQ -194.04  -193.27  -195.00  -194.00  -193.01  -195.47  -193.01
EQ-ADD-MN -194.75  -193.81  -194.06  -196.07  -194.03  -193.08  -194.08

Table 6: Comparison of BICs of models with interleaving-N constraints and blocking constraints. BICs corresponding to models where the coefficient of the predictor is negative are shown in bold.

EQ-MN-ADD is the best way to sequence topics (suggesting that students should at least have some exposure to EQ before MN and some exposure to MN before ADD). This challenges our initial hypothesis that MN-EQ-ADD is the optimal ordering for learning. This result seems to indicate that, in contrast to our hypothesis, learning to make and name fractions (MN) on the number line may be facilitated by knowledge and skill regarding fraction equivalence and ordering (EQ), more so than the other way around. This result may suggest that an understanding of relationships between multiple fractions can help with learning about making and naming individual fractions on the number line, to a greater degree than previously realized. However, we cannot rule out alternative explanations. For example, it could be that our tutor activities are not successful in helping students learn knowledge that transfers to other topics. We note that in the MN activities, students used the number line extensively, whereas they did not in the EQ activities; in the latter they almost exclusively used the symbolic notation of fractions. It may be that if both topics had used the number line, the work on making and naming fractions might have facilitated learning about equivalence and ordering more. Thus, our method for evaluating sequences raises questions about tutor design, which, if and when resolved, could potentially lead to a more effective tutor.

The results on sequencing of activity types were not as unequivocal. We found that the best sequence over activity types may well vary for topics, which is itself an interesting result. For MN and EQ, the models suggest SM should precede F. This result agrees with prior literature on how to order sense-making and fluency activities [14]. However, the relative ordering of SM and IR is not as clear, with it possibly being advantageous to give IR activities before SM activities in many cases, challenging our initial hypothesis.

One may wonder if our results can simply be explained in terms of ordering topics and activity types from easiest to hardest. However, this does not seem to be the case. Note that the performance thresholds in Table 1 provide a measure of difficulty for each topic and each topic-type pair. Based on this measure of difficulty, MN would be classified as easier as EQ, but we saw that our models suggest EQ should come before MN. Furthermore, according to this measure of difficulty, ADD/IR problems would be classified as the most difficult for fraction addition; however, our lowest BIC types models suggest that IR should either come first or second for fraction addition.

Despite the strengths of our method over some prior approaches, the current analysis has several limitations that should be taken into consideration. First, when adaptive problem selection algorithms assign problems to students based on their performance on past problems, the student’s performance can itself impact the proportion of violations of sequencing constraints; thus, SCOVA provides correlational, not necessarily causal, information about the impact of orderings. We can avoid this confound by using data with randomized sequences of problems rather than sequences generated from adaptive policies. However, in many cases (as was the case here) we may not have access to randomly generated sequences, and randomized data can often be difficult to collect ethically if we believe that a random sequence could have negative effects on student learning. To test the degree to which this confounds affects our results, we checked if student’s pretest scores are correlated with the proportion of violations of various sequencing constraints, which would indicate that students with more prior knowledge tend to adaptively be assigned problems that either obey or violate certain sequencing constraints more than students with less prior knowledge. While we did find such correlations for certain sequencing constraints, the coefficients of the pretest score variables used to predict sequencing constraint violations were less than 0.05 in magnitude, and seemed to indicate that higher-performing students tended to receive ADD earlier and EQ later than lower performing students, which is contrary to the sequences we found most predictive of within-tutor performance! Thus we do not think this confound had a worrisome impact on our results.

Second, ideally we would like to see how sequencing constraints impact student learning as measured via posttest scores rather than just within-tutor performance. However, we were unable to find strong correlations between the proportion of violations of sequencing constraints and the posttest scores of students. This is likely due to the fact that the posttest was comprised of only 16 items and as a result is only a noisy measure of a student’s knowledge and does not capture the diversity of concepts taught on the tutor. Note that this is not however a limitation of SCOVA; in theory, SCOVA could be used to compare how various sequencing constraints impact posttest performance.

6. CONCLUSION
We have shown how SCOVA can be used to test a much broader range of sequencing constraints than existing methods (e.g., [13, 21, 19])—including exposure-based, performance-based, interleaving, and blocking constraints. Furthermore, we have shown that when analyzing all of these results in conjunction with each other, a few trends can emerge that can inform practitioners about how to sequence problems. In the case of our fractions tutor, our re-
results suggest presenting students with fraction equivalence before making and naming on the number line, and presenting the latter before fraction addition. In addition, our results suggest that we should not present the topics in a fully blocked fashion, but rather present four problems of each topic at a time. As for activity types, our results suggest that sense-making should typically come before fluency-building, in agreement with prior literature [14], but that the optimal ordering of activity types may vary for certain fractions topics.

These results suggest just some of the use cases of the SCOVA framework. SCOVA can easily be used to test a broader variety of sequencing constraints, as well as informing old debates about sequencing. For example, prior literature has suggested benefits of interleaving in some cases and of blocking in others [2]. From such results, one may be led to wonder “what is the optimal form of interleaving, and under which circumstances?” While it may be difficult to immediately address such a question in an experimental study, due to the sheer size of the space of sequencing constraints, we can easily analyze such a question using SCOVA.

SCOVA can be of benefit to researchers and practitioners in several ways. First, it can lead to refining hypotheses and determining which questions to test empirically (e.g., testing whether EQ should actually precede MN). Second, it can lead to improving the design of tutor problems (e.g., making EQ problems that use the number line and hence build off of the problems that cover making and naming fractions). Finally, it can help with the construction of adaptive policies (e.g., by determining the order of topics in a mastery learning policy as suggested by performance-based constraints).

7. ACKNOWLEDGEMENTS

The research reported here was supported by the Institute of Education Sciences, U.S. Department of Education, through Grants R305A130215 and R305B150008 to Carnegie Mellon University. The opinions expressed are those of the authors and do not represent views of the Institute or the U.S. Dept. of Education.

8. REFERENCES

ABSTRACT
Level creation is a creative game-play exercise that resembles problem-posing, and has shown to be engaging and helpful for players to learn about the game’s core mechanic. However, in user-authoring environments, users often create levels without considering the game’s objective, or with entirely different objectives in mind, resulting in levels which fail to afford the core gameplay mechanic. This poses a bigger threat to educational games, because the core gameplay is aligned with the learning objectives. Therefore, such levels fail to provide any opportunity for players to practice the skills the game is designed to teach. To address this problem, we designed and compared three versions of level creators in a programming game – Freeform, Programming, and Building-Block. Our results show that a simple-to-use building-block editor can guarantee levels that contain some affordances, but an editor designed to use the same core mechanic as gameplay results in the highest-quality levels.

Keywords
User-created Content, Educational Game, Educational Data Mining, Learning Analytics

1. INTRODUCTION
In previous work with our programming game, BOTS, we demonstrated that user-created levels in our game frequently contain appropriate gameplay affordances, which reward specific, desired patterns of gameplay related to the game’s learning objectives. Such levels demonstrate the creator’s understanding of those learning objectives, and offer other players opportunity to practice using those concepts. However, alongside these high-quality submissions there also exist various negative patterns of user-generated content, four of which we specifically defined in previous work: Sandbox, Griefer, Power-Gamer, and Trivial levels. In various ways, these are levels which ignore or replace the game’s core learning objectives and challenges.

Figure 1: Gameplay screenshot from the BOTS game showing a complex puzzle and partial solution.

In order to implement user-created levels into the game itself, an additional filtering and evaluation step is needed to identify and remove these low-quality submission. Our initial attempt at filtering these levels, a “Solve and Submit” procedure, was effective at reducing the number of these types of levels which were published, and additionally was somewhat effective at reducing the number of these levels created to begin with; however, some users created fewer levels under this condition, indicating that the barrier after level creation discouraged further creation. Our next step is to make further improvements to the content authoring tools in order to increase the overall quality of submitted content.
In order to do so, we will investigate three versions of the game’s level editor. The initial, free-form editor, and two constrained editors employing different types of constraints.

Previous work has shown that players are engaged when constraints are posed that are restrictive enough to encourage demonstration of the game’s target learning concepts, but not so restrictive as to require them, lest players feel as though they are unable to create what they want to create. We propose to evaluate level editors with two different forms of constraint added. The Programming Editor, where the length (in lines of code) of the solution is constrained, similarly to the Point Value Showcase in Bead Loom Game. Second, where the construction of the level itself is constrained by providing authors with a limited selection of “Building Blocks”. For this work, we hope to answer (or gain insight into) the question: Does providing game-like scaffolding, in the form of objectives and points related to elements of high-quality content, result in better user authored content?

2. BACKGROUND
User-generated content has been revolutionizing gaming, and the potential applications in educational games are intriguing. Commercial games such as Super Mario Maker\cite{20} and Little Big Planet\cite{19} rely almost entirely on user-submitted levels to provide an extendible gameplay experience, with the creation process itself serving as the meat of the built-in gameplay. Creative gameplay avoids many of the motivational pitfalls of educational games, such as relying on competitive motivators, that may make the intervention less successful for non-males, who may have a more social orientation towards gameplay, or may have less experience with traditional video games\cite{13, 14, 5}.

Creating exercises, in the form of problem-posing, is a common educational activity in many STEM domains. In mathematics in particular, Problem-posing has been promoted as a classroom activity and as an effective assessment of student knowledge\cite{23, 7}. Games and ITSs such as Animal Watch\cite{4} and MONSAKUN\cite{17} have users creating exercises for from expert-selected “ingredients.” Work with systems such as “MONSAKUN”, “AnimalWatch” and the Peer-to-peer learning community “Teach Ourselves” has shown that systems that facilitate problem creation by students can provide benefits beyond those of systems without this feature.

MONSAKUN\cite{17} is a system which facilitates problem-posing for elementary arithmetic problems. The authors wanted to influence students to produce word problems whose structure was different from the structure of the mathematical solution. In order to build the word problem, students are given segments of a word problem such as “Tom has 3 erasers” or “Tom buys several pencils” which they arrange in order to construct their problem.

Animal Watch\cite{1, 4} is a pre-algebra tutor which uses data about exotic animals as the theme for the problems presented. The tutor covers topics such as finding average median and mode, converting to different units, and so on. While the tutor contains around 1000 problems authored by the developers, the authors of this paper noted that even with a large number of problems the system can “run out” of appropriate problems to give a student. The pilot mostly investigated student attitudes towards problem posing, finding that students were excited about sharing content with their peers, and proud that content they had created would be online and accessible to others. At the same time, students reported a low self-assessment of learning, and felt that it was easy once they got started.

Later work by Carole Beal, “Teach Ourselves,” investigated these effects further\cite{3}, incorporating aspects of gamification. Players earn rewards for solving and creating that are displayed on a leaderboard, and can get “+1” from peers for creating good content in the form of problems and hints. Problems created by students were of usable quality, with an average quality score of 7.5/12 on a scale developed by the system’s designers. Teachers who used the system observed increased motivation in their students, and believed that the system encouraged higher-order thinking. Even simple problem-posing interventions have been shown to be effective. In Chang’s work with a problem-posing system to teach mathematics, it was demonstrated that when the posed problems were to be used as content for a simple quiz-show-like game, low performing students experienced significantly greater learning gains from the activity, and students reported being more engaged with the activity\cite{8}.

3. DESCRIPTION OF BOTS
BOTS (bots.game2learn.com) is a puzzle game designed to teach fundamental ideas of programming and problem-solving to novice computer users. BOTS was inspired by games such as LightBot and RoboRally, as well as the syntax of Scratch and Snap\cite{9, 11, 26}. In BOTS, players take on the role of programmers writing code to navigate a simple robot around a grid-based 3D environment. The goal of each puzzle is to press several switches within the environment, which can be done by placing an object (or the robot itself) on top of them. Within each puzzle, players’ scores depend on the number of commands used, with lower scores being preferable. For example, in the first tutorial level, a user could solve the puzzle by using the “Move Forward” instruction 10 times. This is the best score possible without using loops or functions. Therefore, if a player wants to make the robot walk down a long hallway, it will be more efficient to use a loop to repeat a single “Move Forward” instruction, rather than to simply use several “Move Forward” instructions one after the other. These constraints, based on the Deep Gamification framework, are meant to encourage players to optimize their solutions by practicing loops and functions.

Previous work with BOTS focused on how to restrict players from constructing negative design patterns in their levels\cite{16}, and how to automatically generate low-level feedback and hints for user-generated levels without human authoring\cite{22, 10}. Our next steps with this game are to further improve the level authoring tools to increase the quality of the levels which don’t exhibit these negative design patterns.

3.1 Gameplay Affordances
The term Affordance has its origins in psychology, where it is defined by Gibson as “what [something] offers the animal, what it provides and furnishes”\cite{25}. This concept was later introduced to HCI, where Norman defined affordance as “the perceived or actual properties of the thing, primarily those fundamental properties that determine just how the thing
could possibly be used” [21]. Norman’s definition centers on users’ perspectives. If a user does not read an action with an object possible, then the object does not afford that action.

With respect to affordances in games, James Paul Gee wrote that games create “match between affordances and what he calls “effectivities” [12]. In his writing, effectivities are defined as the abilities of the player’s tools in the game; for example a character in a platforming game may be able to run, climb, and jump. On the other hand, affordances describe relationships between the world and actors, or between tools and actors. Other work taxonomizing level design patterns in video games also referred to the desired gameplay produced by these types of structures. For example, in Hullet and Whitehead’s work with design patterns in single-player First-person shooter (FPS) levels, the Sniper Location design pattern is a difficult to reach location with a good view of the play area, occupied by an enemy [18]. This pattern is described as forcing the player to take cover. The presence of other gameplay elements such as Vehicles and Turrets herald similar gameplay changes [2].

In BOTS, the primary educational goal is to teach students basic problem solving and programming concepts such as using functions and loops to handle repetitive patterns. Students (with the robot as their tool) must look at puzzles in terms of opportunities for optimization with loops and functions. Thus, affordances in BOTS come in the form of objects or patterns of objects which both provide and communicate the presence of, these optimization opportunities.

Though the objects in BOTS signal gameplay patterns, players building levels in BOTS frequently place them in misleading or irrelevant ways, where the gameplay decisions informed do not lead to a correct or successful solution. For example, a player can place an extra crate, which communicates that the “Pick Up” command may be used. However, when the optimal solution to the puzzle does not require this crate, the affordance of the crate is meaningless and distracting. Similarly, a player could construct a repetitive structure which affords the use of a “Function” command to navigate, but if ignoring or avoiding the structure entirely results in a better solution, this affordance is also unwanted. Thus, our primary focus is on the subsets of affordances which involve the core mechanisms in question relating to problem solving and solution optimization, and through which players can improve their gameplay outcome in terms of final score. These are referred to as “Gameplay Affordances” in remaining sections.

### 3.2 Level Editors

Specific discussion of the design principles behind the two level editors used for this study can be found in our previous work [15]. For the sake of space, we will only generally discuss those design principles here, instead focusing on the tools available to users in the different designs.

In all versions of the level editor, levels consist of a 10x10x10 grid, where each grid square can be populated by a terrain block or an object. Levels must contain at minimum a start point and goal, and can optionally contain additional goals which must be covered with movable boxes before the level will be completed.

In the Free-Form drag-and-Drop editor, players will be asked to create a level in a Free-Form editor which uses controls analogous to Minecraft. Players can click anywhere in the world to create terrain blocks, and can select objects from a menu such as boxes, start points, and goals, to populate the level with objectives. At any point during creation, the player can save the level (which must, at minimum, contain a start point and a goal.) The player must then complete the level on their own before the level is published and available to other users. In early versions of the Free-form editor, levels began with a 10x10 floor. However, to partially inhibit canvas-filling, this was later changed so that the editor now begins with an entirely blank canvas.

In the Programming Editor (inspired by the Deep Gamification framework [6]) players will be asked to create a level by programming the path the robot will take. To inhibit canvas-filling, players will be constrained to using a limited number of instructions. This is analogous to the level creation tools in BeadLoom Game where players created levels for various “showcases” under similar constraints. This type of constraint has been shown to be effective for encouraging players to perform more complex operations in order to generate larger more interesting levels under the constraints. One challenge with this approach is that since simple solutions are still permitted, and nearly all programs are syntactically correct, users who are experimenting with the level creation interface with no goal in mind may create levels that they themselves do not understand.

In the Building-Block editor, we constrain level creation by providing meaningful chunks to authors in the form of “Building Blocks.” This is inspired by problem-posing activities as presented in systems like MONSAKUN [17] and AnimalWatch [1, 4] in which players are asked to build a problem using data and problem pieces provided by experts. In this version of the level editor, players will be asked to create a level only using our “Building Blocks” which are pre-constructed chunks of levels. These “Building Blocks” will be partial or complete examples of the patterns identified in previous work [15], specific structures which correspond to opportunities to use loops, functions, or variables.
Again, to inhibit canvas-filling, the player is limited to a small number of blocks, regardless of those blocks’ size. We hypothesize that this may lead to better levels because it explicitly promotes the inclusion of these patterns, which will lead to opportunities for players to use more complex programming constructs like loops and functions. We also believe that this will encourage students to think about optimizing the solution to the level while they are making the level. One potential challenge with this approach is that students may find these constraints too restrictive, which might reduce engagement for creatively-oriented players [6]. By evaluating these two versions of a gamified level editor against each other, we will determine which practices best suit our game. In particular, which version of the activity leads to the production of better content for future users.

4. DATA

This paper reports gameplay data from 181 unique user IDs (48 in the Programming condition, 61 using Block Editor, 72 using Free-Form Editor) across all classes/workshops that used the BOTS game as part of their activities. In total, 243 levels were created by these players (91 Block / 59 Programming / 93 Free-Form). Of these levels, 9 Block levels and 6 Programming levels were excluded due to bugs in the early versions of the editors rendering them unplayable after their creation, and 3 additional levels (1 Block level, 1 Programming level and 1 Free-Form level) were removed due to other errors, reducing the total number of levels in the sample to 225 levels (81 / 52 / 92). 175 (49 / 33 / 92) of these levels were published and made public. Additionally, after publication the game continually enforces a minimum ideal solution length of 5, automatically setting levels which meet this criteria to be unplayable. After removing these levels, the final count of levels examined by our zero-inflation model was 197 (73, 44, and 80) puzzles, created by 54, 42, and 64 authors. These participants were participants in STEM workshops organized through SPARCS or other outreach activities. Only anonymized game-play data was used for this analysis, to protect participants. For the Free-Form editor, levels from previous experiments were used, as well as anonymous data from other outreach use of the tool, where the same 90 minute session structure was followed.

The additional data was collected in 90 minute sessions, in which all students followed the same procedure. First, each student created a unique account in the online version of the game. Players then completed the Tutorial up to the final challenge level which functions as sort of a “collector” stage; Players aren’t expected to complete this level with optimum score, but exploring this level allows faster students to continue practicing while the rest of the class catches up. During the tutorial segment, instructors were told to prompt players to reread the offered hints for their current level carefully, if they became stuck, and only to offer more guidance after the player had carefully read the instructions. This part of gameplay took 45 minutes. Data collected with the Free-form editor used an older version of the game with a longer tutorial. We account for this difference between groups by including tutorial completion in our models.

For the remaining 45 minutes, students were instructed to build at least one level in their version of the level editor interface. After collecting this level, players could continue creating levels, or could play levels created by their peers. The way the level editor was selected varied per data collection. In the first set of data collections, (data collected prior to the implementation of the new editors) all students used the “Free-Form” level editor to create their levels. To publish their levels, some students were then required to submit a solution to their level before it became public, however this filtering step took place outside of the level editor and after level creation. Therefore, in this data we make no distinction between published or unpublished levels in this condition. One subsequent data collection used only the “Programming” level editor; this data was initially used to evaluate some graphical elements the interface design of that editor. In the remaining data collections, students were randomly assigned an interface between the “Programming” editor and the “Building-Block” editor.

To analyze the differences between created levels, we played each level to find the shortest-path solution from start to goal, and used a solver to find the shortest program to produce this optimal solution. As the actual process of solving a BOTS puzzle would be as complicated as that of a Light-Bot puzzle [24], we used an algorithm which instead, based on student solutions, finds the best optimization of the shortest discovered path in the level. The algorithm used by the optimization solver is a simple: First, a program that recreates the shortest-path using only simple commands is constructed. Then, sets of repeated commands are identified in this program by treating the commands as words and identifying repeated n-grams. Then, recursively, each possible combination of optimization on these n-grams is applied: either replacing the n-gram with a subroutine identifier wherever it appears, or replacing adjacent -grams with a single instance of that -gram, wrapped in loop commands. After each step, the program is recursively re-evaluated, until the shortest, most optimal version of the solution is found. The shortest-path solution itself is the naïve solution which uses only simple commands such as moving and turning. The optimized shortest-path solution is the expert solution which uses loops and subroutines to optimize the shortest path solution. The difference between these solutions, in terms of lines of code, is used as a measurement of how well the level
affords the use of those game mechanics.

5. METHODS AND RESULTS
In this section, we describe our analyses, both to identify any differences in the presence of gameplay affordances, and to identify differences in how experts tagged the created levels across conditions.

5.1 Overview of level Improvement
In figure 4 we present the box-plot for score improvement between expert and naive solutions. The light and dark-grey sections are a typical boxplot, showing the median and quartiles of the data. From this, we can see that the zero-value levels are certainly over-distributed (especially in the Free-Form condition) which will impact which statistical methods we use to evaluate this measurement. Additionally, the pink area shows the mean value and the 95% confidence interval around it. From visually inspecting this, we can see that these confidence intervals for the Programming Editor and Free-Form editor do not overlap, implying that the Programming Editor achieves better results. We will confirm this with later analysis.

5.2 Expert Tagging
We compared puzzles across three versions of level editors, with the hypothesis that the more meaningful the level editor’s construction unit, the higher quality the puzzles. Here, we assume that “Building Blocks” from version 3 and programs from version 2 are more meaningful than terrain blocks in version 0. We also hypothesize that the Programming editor will result in more reusable puzzles from a player perspective, and that the Building Block editor is more likely to encourage loops and functions.

We used an expert, blind to which editor was used to create the puzzle, tag puzzles, and identify the presence or absence of these negative design patterns. We used the defined puzzle design patterns as identified in our previous work: “Normal” levels which contained few (or no) negative design patterns, and four categories of levels characterized by specific negative design patterns: Griefer, Power-Gamer, Sandbox, and Trivial levels, as described in previous work [16].

We measured a puzzle’s quality based on previously identified patterns of negative content, which were used as tags for this study. The following criteria were used to assign tags:

- a) it is readily apparent that a solution is possible
- b) a solution actually is possible
- c) the solution can be improved with loops or functions
- d) patterns in the level design call out where loops or functions can be used
- e) the expert solution can be entered in reasonable time
- f) the naive solution can be entered in reasonable time

We decided on these criteria because the pedagogical goal of LOGO-like games, such as BOTS, is to teach students basic problem solving and programming skills. Thus, a good quality puzzle should help players focus on the problem, and should encourage the use of fundamental flow control structures like loops and functions. Levels which are impossible, or simply tedious, are among the most common negative traits identified in previous designs, so updated versions of the level editor specifically addressed these two criteria via hard constraints on the placements of goals and size of levels.

Table 1: Categories of Puzzles Created by Three Versions of Level Editors

<table>
<thead>
<tr>
<th>Level Editor</th>
<th>FF</th>
<th>Program</th>
<th>Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>66</td>
<td>43</td>
<td>66</td>
</tr>
<tr>
<td>Power-Gamer</td>
<td>9</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>Griefer</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sandbox</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Trivial</td>
<td>5</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>TOTAL</td>
<td>92</td>
<td>52</td>
<td>81</td>
</tr>
</tbody>
</table>

Table 4 reports the number of puzzles in each category, created by the three level editors. Fisher’s Exact Test showed a significant difference (p<.01) in the category distributions between the three conditions. Each point in this plot represents the difference in number of commands between a naive and expert solution. In this chart, this is represented as a percentage of the expert solution.
between each pair of the three level editors.

The Programming editor has the highest proportion of Normal puzzles. Moreover, the Building Block and Free-Form editors created a higher proportion of Power-Gamer levels compared with the Programming editor. These levels are characterized by extreme length and a high number of objectives. The Free-Form editor is the only level editor in which users created Sandbox puzzles, though since our criteria for Sandbox levels include placing off-path objectives and structures (which is quite difficult in the new editors) this is unsurprising. Finally, the Programming editor has the highest proportion of Trivial puzzles.

Since players in the two new editors used a shorter tutorial than players in the free-form condition, we decided to investigate if student performance in this tutorial had an impact on which level editor was more effective. We considered whether or not the authoring player had completed the new tutorial levels during the allotted time. This analysis is again performed on the reduced data set, with levels with solutions less than 5 steps long removed.

5.3 Direct Measurement of Improvement
To further evaluate the differences between levels on a direct measure of possible improvement (the difference in length between a naive solution and an expert solution) we employed a Zero-Inflation model. This type of model is used for modeling variables with excessive zeros and it is usually for overdispersed count outcome variables. Furthermore, it’s used when theory suggests that the excess zeros are generated by different process from the other values, and can therefore be modeled independently. Our data indeed has an excess of zeroes, due to the measurement in question, number of lines improved, being a minimum of zero. Additionally, in this case, a level with zero improvement contains no affordances, while a level with only a small improvement may still contain affordances that, though present, are less directly rewarding to the player.

Looking at the Poisson model, we see that considering the non-zero results, the Programming editor is likely to have a higher value of Difference than either other condition. In the Building-Blocks editor, each block contains only a small affordance since the blocks themselves are only 3 to 4 commands long. If blocks are not repeated, this pattern will persist in the repeated level. However, in the Programming editor, we observed players exploring more, wrapping code in functions and loops to see what would happen, and changing their code until the level looked how they wanted it to look. Levels generated in this manner will have much larger differences between the naive solutions and expert solutions, than levels generated from multiple unique Building Blocks.

Using a zero-inflated Poisson distribution model, we were able to examine the differences between levels created under our various conditions. We used this zero-inflation model because the model looks for two separate effects: first, the effect that causes the dependent variable to be zero or non-zero, and second, the effect that causes the value of the dependent variable to change in the non-zero cases. This is important because the structural elements for levels with zero affordance for advanced game mechanics are very different from those with only a small affordance—in other words, we would expect the free editor to have more zero-values for the difference between the naive and expert solutions, and the other two editors to have more non-zero values for this difference. Zero-affordance levels tend to be trivially short or entirely devoid of patterns, while small-affordance levels may contain patterns but with small changes between them which limit how advanced game mechanics may be used to optimize the solutions.

To summarize these results, by using this model, we were able to observe the following effects. We first verified our expected result, that both the Programming editor and Building-Block editors are more likely to produce a non-zero result, statistically significantly more likely than the baseline (free-form) condition. The second result is that the Programming editor is likely to have a higher-value difference between and naive and expert solutions, indicating that it promotes puzzles that allow for more optimization.

To investigate if completing the new shorter tutorial had an impact on which level editor was more effective, we considered whether or not the player completed the tutorial levels during the allotted time. The results are presented below:
Table 4: Count model coefficients (poisson with log link) on model, including tutorial completion

|            | Est.  | Std. Err. | z value | Pr(>|z|) |
|------------|-------|-----------|---------|----------|
| (Intercept)| 1.905 | 0.0542    | 35.132  | < 0.001*** |
| Programming| 0.320 | 0.0813    | 3.938   | < 0.001*** |
| Building-Block | -0.060 | 0.078 | -0.769 | 0.442 |
| Tut. Complete | 0.100 | 0.078 | 1.293  | 0.196 |

With this more complex model we see similar results: finishing the shorter tutorial does not have a statistically significant effect, but the coefficient for the magnitude portion of the model is still relatively large. Finishing the tutorial seems to have no compelling impact on the zero portion of the model.

Table 5: Zero-inflation model coefficients (binomial with logit link) including tutorial completion

|            | Est.  | Std. Err. | z value | Pr(>|z|) |
|------------|-------|-----------|---------|----------|
| (Intercept)| -0.568 | 0.233   | -2.436  | 0.015 *  |
| Programming| -1.117 | 0.515    | -2.169  | 0.030 *  |
| Building-Block | -1.083 | 0.424 | -2.556 | 0.011 * |
| Tut. Complete | 0.054 | 0.544 | 0.098 | 0.922 |

To summarize these results, by using this model, we were able to observe the following effects. First, the Building Block editor is most likely to produce a non-zero result, statistically significantly more likely than either other condition. Second, the Programming editor is likely to have a higher-value of difference for the non-zero results that are created.

6. DISCUSSION

The results seem to confirm that the Freeform editor is the least likely to result in levels with gameplay affordances for using loops and functions. The Freeform editor resulted in the lowest proportion of Normal puzzles, but high proportions of Sandbox puzzles and Power-Gamer puzzles. Additionally, they created fewer puzzles that can be improved by loops or functions, or which have obvious patterns for using loops or functions. Players using this editor are less likely to consider the gameplay affordances of their levels, adding elements regardless of their effect on gameplay. Additionally, the Freeform editor is the only level editor where users created Sandbox puzzles. This may be because Sandbox levels are characterized by the presence of extraneous objects, and the new editors operate by creating the robot’s path, so designers would have to deliberately stray from their intended path to place extraneous objects.

On the other hand, the Programming Editor resulted in a high proportion of Normal puzzles and the lowest proportion of Power-Gamer puzzles. This makes sense because a Power-Gamer puzzle is typically a puzzle which takes a short time to create but a long time to complete. Since this editor uses the exact same mechanic for creation as completion, this is quite difficult to do. However, these users also built a lower proportion of puzzles that can be improved with loops and functions than the users of the Building Block editor, and the highest proportion of Trivial puzzles whose solutions are too short to afford the use of loops or functions. This editor is the most complex to use, so players with little patience for learning the interface may create Trivial puzzles. Additionally, trying options at random to see what they do in the programming editor is likely to result in the creation of a Trivial level. We hypothesize that in the other editors, random behavior results in different level types: Power-Gamer levels in the Building Block editor, and Trivial levels in the Programming editor.

Lastly, the Building-Block Editor has a high proportion of normal puzzles, and is slightly more likely to generate a non-zero result than the Programming editor. The building blocks used to create levels are subsections of previously created levels selected specifically because they afford the use of loops or functions. The Building-block editor created the highest proportion of Power-Gamer puzzles. This may be because of the ease of use; adding a block takes one click but may require 5–10 commands from the player who later solves the puzzle. We previously observed that players tended to fill the space available to them in the Freeform editor, so Building-block puzzle creators may also be trying to fill the available space. In both other editors, it takes longer to solve the puzzle than to create it, but the programming editor minimizes this difference, thereby making the creation of Power-gamer levels less likely.

7. CONCLUSIONS AND FUTURE WORK

In conclusion, including Deep Gamification elements in Level Editors (in the form of creative constraints, building blocks, or integration with gameplay mechanic) did result in an overall improvement in level quality. In both the Programming editor and Building-Block editor were more effective than a Freeform editor at encouraging the creation of levels which contain gameplay affordances. The Programming editor was most effective at ensuring a non-zero improvement between expert and naive solutions, but perhaps trivially so, as the building blocks themselves were selected as to contain small improvements. The Programming editor is less likely to ensure a non-zero improvements, but levels created under this condition contain larger improvements, which may be more obvious or more rewarding to players than numerous small improvements.

Our next steps are to investigate how players react to levels created under these conditions. We know that these levels contain opportunities for users to practice, but if the users don’t recognize or simply don’t take advantage of the opportunities, the improvement is lost. Additionally, we noticed several patterns of negative design that are unique to these new editors, with regards to canvas-filling behaviors. This results in shifting “Sandbox” design into Power-Gamer or Trivial levels. For the new editors, this seems to be mostly negative, resulting in overlong, unrewarding levels. However in the Programming editor, this behavior sometimes resulted in interesting levels created when the author was experimenting with loops and nested functions rather than creating with an end-goal in mind. Similar experimental usage of the previous level editor was treated as negative, with the output levels being low-quality. In the Programming editor, that is not always the case, so re-evaluation of how these levels are identified is needed.
8. ACKNOWLEDGMENTS
Thanks to Michael Kingdon, Aaron Quidley, Veronica Catete, Rui Zhi, Yihuan Dong, and all developers who have worked on this project or helped with our outreach activities so far. This project is partially funded under NSF Grant Nos. 0900860 and 1252376.

9. REFERENCES
The Eyes Have It: Gaze-based Detection of Mind Wandering during Learning with an Intelligent Tutoring System

Stephen Hutt, Caitlin Mills, Shelby White, Patrick J. Donnelly, & Sidney K. D'Mello
University of Notre Dame
384 Fitzpatrick Hall, Notre Dame, IN, 46556, USA
{shutt, cmills4, swhite16, pdonnel4, sdmello}@nd.edu

ABSTRACT

Mind wandering (MW) is a ubiquitous phenomenon characterized by an unintentional shift in attention from task-related to task-unrelated thoughts. MW is frequent during learning and negatively correlates with learning outcomes. Therefore, the next generation of intelligent learning technologies should benefit from mechanisms that detect and combat MW. As an initial step in this direction, we used eye-gaze and contextual information (e.g., time into session) to build an automated MW detector as students interact with GuruTutor – an intelligent tutoring system (ITS) for biology. Students self-reported MW by responding to pseudorandom thought-probes during the tutoring session while a consumer-grade eye tracker monitored their eye movements. We used supervised machine learning techniques to discriminate between positive and negative responses to the probes in a student-independent fashion. Our best results for detecting MW (F₁ of 0.49) were obtained with an evolutionary approach to develop topologies for neural network classifiers. These outperformed standard classifiers (F₁ of 0.43 with a Bayes net) and a chance baseline (F₁ of 0.19). We discuss our results in the context of integrating MW detection into an attention-aware version of GuruTutor.

Keywords

eye-gaze, intelligent tutoring systems, mind wandering, attention-aware learning

1. INTRODUCTION

It is safe to say that most of us have had the experience of reading a text or listening to a lecture and then suddenly realizing that our thoughts have drifted to completely unrelated things, such as an upcoming vacation. This phenomenon, known as mind wandering (MW), refers to the unintentional shift of attention away from the current task towards internal task-unrelated thoughts [32]. MW is a ubiquitous phenomenon, estimated to occur as much as 50% of the time depending on the individual, task, and environment [16].

Not only does MW occur frequently, it can have detrimental influences on performance, especially during educational activities. Indeed, a recent meta-analysis revealed a negative correlation between MW and performance across a variety of tasks, such as lower recall in memory tasks and poor comprehension in reading tasks [24]. It is prudent to point out that MW is not always harmful and the tendency to day-dream has been shown to aid in certain types of tasks, such as creative problem solving [20]. However, research consistently shows that MW impairs performance in tasks requiring concentrated attentional focus and integration of information from the external environment as is the case with many learning activities [21].

Considering the negative influence of mind wandering on learning [27, 29, 30], it is important to take steps towards developing intelligent systems that help reorient attention to assure the negative effects of MW. This requires an ability to monitor the locus of attention, detect students’ current attentional state, and provide a stimulus to direct focus back to the learning task [10]. Detecting MW is no easy task however. Although MW is related to other forms of disengagement, such as boredom, behavioral disengagement, and off-task behaviors [1, 2, 9, 18, 36], it is inherently distinct because it involves internal thoughts rather than overt expressive behaviors. This raises two challenges. First, while other disengaged behaviors often involve detectable behavioral markers (e.g., yawns signaling boredom), mind wandering is an internal state that can look similar to on-task states. Secondly, the onset and duration of MW cannot be precisely measured because MW can occur outside of conscious awareness.

Despite these challenges, there has been some progress toward automatic detection of mind wandering during reading (discussed as related works in Section 1.1). However, almost all of the current MW detectors focus on reading tasks, so their effectiveness is unclear during complex interactive tasks, such as learning with advanced learning technologies. Here, we explore for the first time, automated approaches for MW detection during learning with intelligent tutoring systems (ITS).

1.1 Related Work

In an early study attempting to detect MW in the context of learning [11], students were asked to read a paragraph about biology aloud, followed by either self-explanation or paraphrasing. Students self-reported how frequently they zoned out on a scale from 1 (all the time) to 7 (not at all). A supervised machine learning model trained on acoustic-prosodic features to classify low (1-3 on the scale) and high (5-7 on the scale) zone outs achieved an accuracy of 64%. However, it is unclear whether this detector could generalize to new students as the validation method did not ensure student-level independence across training and testing sets.

Some researchers have built MW detectors based on information readily available in log files collected during the reading (e.g., reading time, complexity of the text). For example, [19], attempted to classify whether students were MW while reading a screen of text using reading behaviors and features of the text,
such as text difficulty. They were able to classify MW at 21% greater than chance using a leave-one-subject out cross-validation method. Similarly, another study [12] also attempted to predict MW during reading using textual features, such as word familiarity, difficulty, and reading time. However, rather than using supervised machine learning, they used a set of researcher-defined thresholds to ascertain if participants were “mindlessly reading” based on difficulty and reading time.

More recent studies have explored additional techniques to detect MW during self-paced computerized reading [5, 7, 12, 19]. In these studies, MW was measured via thought probes that occurred on pseudo-random screens (i.e. screen of text similar to a page of text). Participants responded either “yes” or “no” based on whether they were MW at the time of the probe. Supervised classification models were trained to discriminate the two responses using physiological features (e.g., skin conductance, temperature) [7] or eye-gaze [9], achieving accuracies ranging from 18% to 23% above chance and validated in a manner that generalized to new students. Further, combining the two modalities led to a 11% improvement in detection accuracy above the best individual modality [3].

Previous attempts to detect MW from eye-gaze are of particular relevance to the current paper. Eye tracking offers a unique possibility to automatically detect MW due to well-known relationships between visual attention and eye-movements. For example, MW has been associated with longer fixation durations [26] and more blinking in reading [33]. These and other relationships have been leveraged to build MW detectors during reading [4, 6] with moderate levels of success. However, it is unclear if these findings and corresponding detectors generalize to other activities, particularly activities where eye-gaze does not have the predictable patterns found in reading text.

1.2 Current Study and Novelty

The primary focus of this paper is to detect MW during learning with an ITS called GuruTutor. Previous work suggests that MW occurs, on average, once every two minutes during interactions with GuruTutor and is negatively correlated with learning gains [17], highlighting the importance of detecting MW in this context.

There are a number of novel aspects with this work. First, we study MW detection in an interactive context— an ITS with conversational dialogues and other embedded activities. Detection of MW during interactions with an ITS provides additional challenges compared to reading. In reading tasks, it is generally clear where the reader should be looking if they are engaged in the task and the eyes move across the screen in a predictable manner. However, in complex environments such as an ITS, there are far more paths the eyes may take, resulting in fewer predictable patterns, rendering MW detection more difficult.

Second, GuruTutor includes multiple activities, such as lecturing, scaffolded dialogue, concept mapping, and Cloze task completion. Each has a different visual layout, level of interactivity, and learning goal, presumably engendering different gaze patterns and levels of MW. By requiring our MW detector to work across a range of activities, we hope to have a solution that will generalize to additional learning technologies that may support quite different activity types.

Third, while researchers have typically used standard classification algorithms (e.g., Naïve Bayes, decision trees), we explore the use of a genetic algorithm (GA) to evolve neural networks (both topologies and connection weights) for detecting MW. This approach evolves the weights and topology concurrently, thereby implicitly integrating feature selection and feature weighting. Further, MW detection suffers from a data-imbalance problem in that the standard classifiers are skewed towards predicting the majority class, which is typically the class associated with Not MW. We address this issue by considering various GA fitness functions that focus on balancing the precision and recall of the minority MW class.

Fourth, we use a low-cost consumer-grade eye tracker to collect gaze data from participants as they interact with Guru. Research grade eye trackers can cost upwards of $40,000, so the use of affordable equipment (less than $150) increases the scalability of the detector for eventual deployment in real world learning environments such as computer-enabled classrooms.

2. DATA COLLECTION

We adopted a supervised classification approach for MW detection, which entailed collection of training and validation data.

2.1 Participants

Participants were 105 undergraduate students (69.5% female, average age 19.14) from a mid-sized, private university in the Midwest. Participants received extra credit or course credit for participating in the study.

2.2 GuruTutor

GuruTutor (Guru) is an ITS designed to teach biology topics through collaborative conversations in natural language. It is modeled after interactions with expert human tutors [22]. Guru engages the student through natural language conversations with an animated tutor agent that references a multimedia workspace, animating content relevant to the conversation (see Figure 1). Students type in responses in a conversational style that Guru analyzes using natural language processing. Guru maintains a student model which it uses to tailor instruction to individual students. Guru has been shown to be effective at promoting learning and retention at levels similar to human tutors [22].

![Figure 1. Example of Guru during CGB Phase](image-url)

Guru presents biology topics aligned with state curriculum standards (e.g., cellular respiration), typically lasting between 15 to 40 minutes each. Each topic contains sets of interrelated concepts and facts (e.g., proteins help cells regulate functions). Guru begins each new topic with a brief preview to introduce it to the student, followed by a five phase session that encourages students to build and articulate their understanding of the concepts. These five phases are described below.

Common-Ground-Building Instruction (CGB Instruction). Biology lessons often involve specialized terminology that needs to be well understood before it is possible to move on to more collaborative knowledge building activities. Therefore, Guru...
begins with a collaborative lecture phase that covers basic information and terminology relevant to the topic. **Intermittent Summaries (Summary).** Following CGB, students generate summaries using natural-language to describe the content covered. These summaries are automatically analyzed to determine which concepts to target throughout the remainder of the session. **Concept Maps.** For the target concepts, students complete skeleton concept maps, node-link structures that are automatically generated from concept text. **Scaffolded Dialogue.** Next students complete a scaffolded natural language dialogue in which GuruTutor uses a Prompt → Feedback → Verification Question → Feedback → Elaboration cycle to cover target concepts. If a student shows difficulty mastering particular concepts, a second Concept Maps phase is initiated followed by an additional Scaffolded Dialogue phase. **Cloze Task.** The session concludes with a cloze task requiring students to complete an ideal summary of the topic by filling in blanks to connect key words to related concepts.

![Figure 2. Example of Guru during Concept Maps](image)

### 2.3 Procedure

All experimental procedures were reviewed and approved by the university’s ethics board. After signing an informed consent, participants were seated at a desk in front of a 15-inch laptop. A Tobii EyeX eye-tracker was positioned directly under the laptop screen using a magnetic strip based on the guidelines provided by Tobii.

Participants were asked to sit comfortably with the chair pulled up to the desk. Next, participants were given an explanation of MW and were given detailed instructions for how to respond to the mind wandering probes (see below) during learning with Guru. Specifically, MW was defined as “when you realize that you are no longer paying attention to what you’re supposed to be doing, for example, instead of thinking about the biology, you may be thinking about something else altogether.”

After receiving initial instructions, a 60 second calibration process occurred before beginning the learning session. Participants were dynamically instructed about their seating and head position in order for the eye tracker to pick up their eye gaze.

Then, one of six biology topics from Guru was assigned to each participant: Interphase, Osmosis, Biochemical Catalysts, Carbohydrate Function, Protein Function, or Facilitated Diffusion. Following a pretest on the assigned topic, participants began the Guru tutoring session. Afterwards, participants completed a posttest and were fully debriefed.

### 2.4 Mind Wandering Probes

Mind wandering was measured during learning with Guru using auditory thought probes, which is a standard approach in the literature [31]. Participants were probed at pseudo-random intervals with probes occurring every 90-120 seconds, this was based on previous work investigating how often MW occurs[17]. If the tutor was speaking at the time the probe was triggered, the probe was paused until the tutor finished speaking so as to not interrupt the conversation flow. Probes consisted of an auditory beep that automatically paused the tutoring session. An opaque overlay would then appear on screen, instructing the participant to press the “N” key if they were not mind wandering, the “I” key if they were intentionally (deliberately) mind wandering, or the “U” key if they were unintentionally (spontaneously) mind wandering. In this study, we do not differentiate between intentional and unintentional mind wandering, and “I” and “U” responses were coded as “MW” to indicate mind wandering occurred. Participants encountered an average of ten probes over the course of the session. We obtained a total of 1104 reports to thought probes, 17% of which corresponded to episodes of MW.

### 3. MODEL BUILDING

Supervised machine learning models were built to detect MW using eye-gaze data and contextual information from Guru.

#### 3.1 Feature Engineering

We calculated features from a short window of time preceding each auditory probe, exploring window sizes ranging from 3 to 30 seconds. We did not consider windows shorter than 3 seconds, as they most often did not contain sufficient gaze data. We discarded windows where not all the eye-gaze features could be computed, such as cases when the face was occluded or the student was looking down at the keyboard. For the smallest window (three seconds) 418 instances were removed, lowering the MW rate to 15.5%. A total of 156 instances were removed for all other window sizes, leaving the average MW rate unaffected (17%).

<table>
<thead>
<tr>
<th>Table 1. Eye-gaze features</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fixation Duration</strong></td>
</tr>
<tr>
<td><strong>Saccade Duration</strong></td>
</tr>
<tr>
<td><strong>Saccade Length</strong></td>
</tr>
<tr>
<td><strong>Saccade Angle Absolute</strong></td>
</tr>
<tr>
<td><strong>Saccade Angle Relative</strong></td>
</tr>
<tr>
<td><strong>Saccade Velocity</strong></td>
</tr>
<tr>
<td><strong>Fixation Dispersion</strong></td>
</tr>
<tr>
<td><strong>Horizontal Saccade Proportion</strong></td>
</tr>
<tr>
<td><strong>Fixation Saccade Ratio</strong></td>
</tr>
</tbody>
</table>

*Note. Bolded cell indicates that the total number, mean, median, min, max, standard deviation, range, kurtosis, and skew of the distribution of each measurement were used as features.*

**Gaze Features.** Eye movements are measured by fixations (i.e. points in which the gaze was maintained on the same location) and saccades (i.e. the movement of the eyes between fixations). We calculated fixations and saccades from the raw eye-gaze data using the Open Gaze and Mouse Analyzer (OGAMA) [35], an open source package for eye tracking analysis. Next, gaze features were computed for each from the fixations and saccades (see Table 1) in that window. We considered six general measures based on fixations and saccades. For these gaze measures, we calculated the number, mean, median, min, max,
standard deviation, range, kurtosis, and skew of the distributions of each measure across the time window, yielding 54 features. We also included three other features (listed in Table 1), yielding a total of 57 gaze features.

**Contextual Features**. The gaze features were complemented with eight contextual features that provide a snapshot of the student-tutor interaction context during each window. One feature was the assigned biology topic. A second encoded participants’ pretest scores on that topic. The next three of these features describe participants’ progress within Guru, such as the current phase of the session (e.g., cloze, concept map, etc.), the amount of elapsed time into the session, and the amount of elapsed time into the current phase. The last three context features focused on participants’ overall interaction with Guru, measured by the amount of positive, neutral, and negative feedback received.

### 3.2 Addressing Class Label Imbalance

Only 17% of the 1104 thought probes were reports of MW, thereby leading to substantial data skew. This imbalance between the class labels poses a challenge as some supervised learning methods tend to bias predications towards the majority class label. To compensate for this concern, synthetic oversampling was applied to provide a more balanced class distribution on the training set only. The SMOTE algorithm [8] creates synthetic instances of the minority class by interpolating feature values between an instance and randomly chosen nearest neighbors. No SMOTING was done on the testing set in order to ensure validity of the predictions.

### 3.3 Classification Models

We evaluated five classifiers frequently explored for the detection of MW [6, 7]. These included Bayesian networks, logistic regression classifiers, multilayer perceptrons (MLP), random forests, and support vector machines (SVM) using implementations from the WEKA data mining software [14].

We also considered a neural network trained using a genetic algorithm (GA), which is a type of evolutionary algorithm for optimization and search problems that uses techniques loosely inspired by biological natural selection. GAs maintain a population of candidate solutions (phenotypes), each with a set of properties (genotypes). These individual solutions evolve over time guided by a fitness function. At each generation, the fitness function is used to rank the candidate solutions, allowing elimination of inferior solutions and selection of the best candidates to the new generation. New candidate solutions are created at each generation through the mechanisms of mutation, a pseudo-random perturbation of an individual’s genotype, and cross-over, the combination of aspects of the genotypes of multiple fit individuals.

**NEAT Algorithm**. In this study, we used a GA to evolve an artificial neural network for MW detection. We used the NeuroEvolution of Augmenting Topologies (NEAT) algorithm to evolve the topology of neural network alongside an evolution of the network weights [34]. Because NEAT evolves both the weights and topology of the network, it must implement the genetic operators of mutation and crossover in a unique way to handle differences between network topologies. NEAT uses population speciation to track individuals with similar topologies, restricting crossover to individuals with similar network topologies to ensure the resulting new topology is coherent. Mutation of the topology occurs in two ways, either by the creation of a hidden node or the addition or removal of a link between nodes. As the size of the networks may grow larger in each new generation, constraints are imposed to penalize large networks that exceed a complexity threshold.

To encourage innovation in new generations, NEAT implements speciation by grouping networks that share similar topologies into the same population. The populations are determined by a distance metric that computes the distance of a topology of an individual from the initial topology of the species. New populations are created as new networks that are dissimilar from any existing population evolve. This strategy allows the generation of new individuals by applying genetic operators on similar individuals in order to maintain viable network topologies without hindering the ability of the GA to develop new and unique networks.

**Using NEAT for MW Detection**. We used SharpNeat, a popular implementation of the NEAT algorithm in the C# language [28]. We tuned the evolution variables on our data in preliminary experiments. We used a population of 150 individuals and ran the algorithm for 500 generations. We also determined a complexity threshold to prune overly complex networks. Because evolutionary algorithms are non-deterministic, we ran these classifiers over multiple iterations in each experiment.

The effectiveness of an evolutionary algorithm depends on the evaluation of individuals using the fitness function. We considered three different fitness functions that were informed by [13]. The first function evaluates candidate networks using the overall accuracy (recognition rate) of the model. The second function evaluates the networks considering the F1 measure for the class label of interest, which in our case is MW (denoted as F1-MW). The third evaluates the networks using the Youden’s J-statistic, (a variation on Cohen’s Kappa, sometimes called “informedness” [23]) which is defined as sensitivity + specificity – 1 of MW.

### 3.4 Cross-Validation

All experiments were conducted using leave-several-participants-out cross-validation. For each iteration of the classifier, instances from 66% of the participants were assigned to a training set and the remaining instances of the other 33% participants were assigned to a test set. This process ensures that no instances of any individual participant could appear in both the training and test sets within a fold. This process was repeated for 15 folds, and the results accumulated. We selected 15 iterations in order to balance time taken to build the models (as evolutionary approaches are slow) and reliability by testing multiple training/testing set pairs. Minority oversampling (SMOTING) occurred within each fold and on the training set only.

### 4. RESULTS

We report the F1 measure in our evaluation of our results. This measure is common in information retrieval tasks and provides a balance between precision and recall. Because our intention is to detect instance of MW, we focus on the F1 score of the MW label as our key metric. This is a very strict evaluation criterion as the base rate of MW is only 17% in our data. To facilitate comparisons with previous (and future work), we also reported the F1 score for the majority (Not MW) class (83% of instances), as well as the weighted F1 score.

#### 4.1 Comparing Window Size

In our first experiment, we explored the influence of various window sizes ranging from 3 to 30 seconds. As we are interested
in general trends, we average results of the five standard classifiers and the three NEAT classifiers. (see Figure 3). These results illustrate a general trend of improved performance for the larger windows, although these differences may not be overly large. In the remainder of this work, we considered a 30 second window in our experiments as it generally resulted in the highest F1 scores.

![Figure 3. Comparison of different window sizes.](image)

4.2 Comparison of Classifiers

In Table 3 we report the results of the classifiers considering a 30 second analysis window, informed by our experiment in Section 4.1. The highest F1 for MW is denoted in bold for both the common classifiers and NEAT implementations that varied by fitness function. For comparison, a chance-level baseline was created by randomly assigning a class label to each instance based on the observed MW rate of 17%. We note that all of the classifiers showed an improvement in detecting the target minority class of MW over the chance model.

Table 2. MW detection results by classifier for 30 second window

<table>
<thead>
<tr>
<th>Classifier</th>
<th>F1 of MW</th>
<th>F1 of Not MW</th>
<th>Overall F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Classifiers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bayesian Network</td>
<td>0.43</td>
<td>0.73</td>
<td>0.68</td>
</tr>
<tr>
<td>Logistic</td>
<td>0.38</td>
<td>0.79</td>
<td>0.72</td>
</tr>
<tr>
<td>Regression</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLP</td>
<td>0.30</td>
<td>0.83</td>
<td>0.74</td>
</tr>
<tr>
<td>SVM</td>
<td>0.37</td>
<td>0.76</td>
<td>0.70</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.23</td>
<td>0.86</td>
<td>0.75</td>
</tr>
<tr>
<td>NEAT Classifiers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fitness: Accuracy</td>
<td>0.36</td>
<td>0.76</td>
<td>0.69</td>
</tr>
<tr>
<td>Fitness: F1-MW</td>
<td>0.49</td>
<td>0.58</td>
<td>0.57</td>
</tr>
<tr>
<td>Fitness: Youden J</td>
<td>0.44</td>
<td>0.69</td>
<td>0.65</td>
</tr>
<tr>
<td>Baseline</td>
<td>0.19</td>
<td>0.83</td>
<td>0.73</td>
</tr>
</tbody>
</table>

Among the common classifiers, Bayesian network achieved the highest F1 score for MW. This was also the case in previous MW eye-gaze detectors in other domains [6]. The overall F1 score for the Bayesian network was lower than for other classifiers, ostensibly because the other classifiers tend to over predict the majority class. For NEAT, using the F1-MW score as the fitness function resulted in the overall best F1 score for MW. NEAT with Youden’s J-statistic as the fitness function did yield a slightly more balanced detector with an increase in F1 of Not MW. Importantly, the best NEAT classifier outperformed the Bayesian network at detecting MW, which is our target class of interest. In Table 3 we show the confusion matrices for the three classifiers that obtained the highest F1 score for MW: the Bayesian network, NEAT-F1-MW, and NEAT-Youden. NEAT-F1-MW yielded a substantially higher hit rate than the other two classifiers, but also suffered from a high false positive (FP) rate. The Bayesian network and NEAT-Youden had similar patterns of errors in that they had both lower hit rates as well as FP rates. Based on these results, we consider NEAT-F1-MW and the Bayesian network in subsequent analyses.

Table 3. Confusion matrices for the three best classifiers

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Actual MW</th>
<th>Predicted Not MW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayes Net</td>
<td>MW</td>
<td>Not MW</td>
</tr>
<tr>
<td>MW</td>
<td>0.52</td>
<td>0.48 (miss)</td>
</tr>
<tr>
<td>Not MW</td>
<td>0.34 (false pos.)</td>
<td>0.66 (correct rej.)</td>
</tr>
<tr>
<td>NEAT-F1-MW</td>
<td>MW</td>
<td>Not MW</td>
</tr>
<tr>
<td>MW</td>
<td>0.69</td>
<td>0.31 (miss)</td>
</tr>
<tr>
<td>Not MW</td>
<td>0.54 (false pos.)</td>
<td>0.46 (correct rej.)</td>
</tr>
<tr>
<td>NEAT-Youden</td>
<td>MW</td>
<td>Not MW</td>
</tr>
<tr>
<td>MW</td>
<td>0.55</td>
<td>0.45 (miss)</td>
</tr>
<tr>
<td>Not MW</td>
<td>0.41 (false pos.)</td>
<td>0.59 (correct rej.)</td>
</tr>
</tbody>
</table>

4.3 Gaze only vs. Gaze + Context Features

We investigated the utility of contextual features over the gaze features alone (see Table 4). The addition of contextual features improved the F1 score for the minority class of MW for NEAT and correspondingly for the majority Not MW class for the Bayesian network. Overall, the improvements in performance were small, suggesting that the gaze features were more important to the detection of MW compared to the contextual features.

Table 4. Gaze (G) vs. Gaze + Context (G+C) features

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Feature</th>
<th>F1 of MW</th>
<th>F1 of Not MW</th>
<th>Overall F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayesian network</td>
<td>G</td>
<td>0.45</td>
<td>0.69</td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td>G+C</td>
<td>0.43</td>
<td>0.73</td>
<td>0.68</td>
</tr>
<tr>
<td>NEAT-F1-MW</td>
<td>G</td>
<td>0.44</td>
<td>0.58</td>
<td>0.56</td>
</tr>
<tr>
<td></td>
<td>G+C</td>
<td>0.49</td>
<td>0.58</td>
<td>0.57</td>
</tr>
</tbody>
</table>

4.4 Oversampling vs. No Oversampling

In Section 3.2, we discussed the imbalance between instances of MW and Not MW in the dataset, and addressed this difficulty by supplementing the training data with the SMOTE oversampling technique. To study the effect of SMOTE, we compared the Bayesian network and the best NEAT classifier on datasets with and without these synthetic training instances (see Table 5). We confirmed that synthetic oversampling indeed improved the classification of the MW (the minority class) for NEAT at the cost of detecting the majority class. Thus, SMOTING played a critical role in reducing the tendency to over predict to the majority class. SMOTING had no notable effect for the Bayesian network, which seemed to be more impervious to data skew.
Table 5. Results with and without oversampling.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>SMOTE</th>
<th>F1 of MW</th>
<th>F1 of Not MW</th>
<th>Ove all F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayesian net</td>
<td>No</td>
<td>0.41</td>
<td>0.75</td>
<td>0.70</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.43</td>
<td>0.73</td>
<td>0.68</td>
</tr>
<tr>
<td>NEAT-F1-MW</td>
<td>No</td>
<td>0.42</td>
<td>0.75</td>
<td>0.79</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.49</td>
<td>0.58</td>
<td>0.57</td>
</tr>
</tbody>
</table>

4.5 Analysis of Features

Neural networks use a mathematical approach to transform and combine input features to useful output. Thus, we can learn more about the structure of our MW detector by investigating the topologies formed during the evolutionary process. For example, a network with a densely connected hidden layer would be performing a large amount of internal calculations compared a sparsely connected layer.

To better understand our MW detector’s structure, we examined each of the 15 iterations of the NEAT-F1-MW model and investigated the networks that survived to the final generation in each case. Across the networks the mean number of hidden nodes in the network is 1.6 (min 0, max 3), the average number of inputs actually used in the final network is 17.133 (min 8, max 36) and the average number of connections is 21.46 (min 9, max 44). The number of hidden nodes here is low, but considering the large number of inputs to a small number of outputs, this is to be expected. The algorithm also biases towards smaller networks to avoid bloat.

Developing neural network topologies also provides inherent feature selection that takes place as the network structures evolve to subsequent generations. This provides an opportunity to explore which features were most useful in detecting MW. Seven features appeared in at least half of the final networks as shown in Table 6.

Table 6. Cohen’s d of most commonly used features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Cohen’s d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixation Duration Skew</td>
<td>-0.27</td>
</tr>
<tr>
<td>Minimum Fixation Duration</td>
<td>0.17</td>
</tr>
<tr>
<td>Mean Saccade Duration</td>
<td>0.32</td>
</tr>
<tr>
<td>Saccade Duration Kurtosis</td>
<td>-0.16</td>
</tr>
<tr>
<td>Saccade Duration Skew</td>
<td>-0.17</td>
</tr>
<tr>
<td>Minimum Saccade Velocity</td>
<td>-0.15</td>
</tr>
<tr>
<td>Fixation to Saccade Ratio</td>
<td>-0.17</td>
</tr>
<tr>
<td>Pre Test Score</td>
<td>-0.18</td>
</tr>
</tbody>
</table>

We compared these seven features across the MW and not MW instances using an effect size measure (Cohen’s d). An effect size measure is appropriate for this comparison in order to evaluate the direction and magnitude of the differences between the two classes. Positive values depict higher values for instances of MW (see Table 6). In general, the differences reported in this paper are consistent with previous work examining eye gaze surrounding MW episodes during reading [4]. Two of the seven features had differences across the MW and not MW classes consistent with small effect sizes (|d| > .2). The largest difference was seen for mean saccade duration (d = .32). This finding suggests that participants tend to have longer gaps between fixations leading up to a MW episode as opposed to more rapid eye movements between fixations. A similar effect size was found for fixation duration skew (d = -.27), which suggests that there is a higher probability that participants would have shorter fixations before a MW episode occurs compared to when their attention is on task.

It is important to point out that the low Cohen’s d values (< .2) are not entirely surprising given the nature of neural networks. The network employs a combination of features and the combination sets that prove to be most effective for MW detection may not be consistent with the overall largest mean differences. Instead, the important thing to note is that these seven features were the most consistent across all iterations.

It is also worth mentioning that only one context feature was present in over half of the final networks: pre-test score. Instances of MW were associated with lower pre-test scores, indicating that when participants were more likely to mind wander if they did not understand the topic well to begin with.

5. GENERAL DISCUSSION

Mind wandering occurs frequently during learning and has a negative impact on learning outcomes [21]. An attention-aware learning technology [10] that can automatically detect MW could intervene to re-engage learners, assuring the cost of MW on comprehension to improve learning. However, MW is a covert, internal state with no obvious behavioral markers, making it difficult to detect. Although strides have been made to detect MW in the context of self-paced reading, MW detection has not yet been attempted in the context of an ITS – a challenge we addressed in the current paper. In the remainder of this section, we discuss our main findings, consider potential applications, and discuss limitations and future work.

5.1 Main Findings

MW detection during reading tasks is supported by decades of research on MW and eye movements [25]. However, more complex learning interfaces, such as the ITS used here, are not afforded such predictable patterns of eye movements. Despite these challenges, we demonstrated the ability of a neural network trained using a GA to detect MW in the context of learning with an ITS. We were able to accurately classify MW with an F1 of 0.49 at detecting the minority MW class. Although this result is modest, it is an important first step in detecting MW in this novel domain.

In most machine learning tasks, a large imbalance in the distribution of class labels results in a degraded performance at predicting the minority class label [15]. This is a major issue for MW detection as its rate of occurrence is around 20% to 40% in learning contexts [27] and in our case it was 17%. We addressed the data imbalance by using a synthetic oversampling technique and by tweaking the fitness function of the GA in order to help the classifiers in detecting the minority class of MW. We believe that this combined approach might be beneficial for other classification problems when there is severe data skew.

Since MW detection in the context of learning from an ITS is still in its infancy, it was important for us to adopt a method that will generalizable for future work in this area. The eye gaze feature set was limited to eye movements that were independent of the specific content being displayed on the screen. This enabled our models to operate across Guru’s multiple instructional activities, each with very different visual displays.

In addition to the gaze features, a second set of features included the context of the learning session. A comparison of model performance with and without contextual features revealed that contextual features added a small, but not substantial, improvement in detection accuracy. This finding further illustrates the idea that eye gaze can be a powerful signal of attention, regardless of the learning context.
An analysis of the most consistent features in the model point to seven important features, six of which are gaze features. MW episodes had a longer mean saccade duration, yet smaller fixation duration skew. The longer mean saccade duration preceding MW is consistent with prior research, which suggests that MW signals a breakdown at very basic levels of perceptual processing [30] – in this case, being slower to direct your eyes form one point to another. Most of the effect sizes (d’s) are objectively small effects; however, we feel that obtaining a sense of consistent features and how the relate to MW is a major contribution at this stage in the of MW detection.

All data was collected using low-cost, consumer-grade eye trackers (less than $150). This is a marked contrast compared to many research-grade trackers that can cost tens of thousands of dollars. Our goal is eventual deployment of our models at scale, thereby allowing us to test generalizability in more diverse contexts. For this reason, it was important to ensure that our models were validated in a student-independent manner, which increases our models’ ability to generalize to new students. Taken together, these results increase our confidence that the models will generalize more broadly, though this claim requires further empirical validation.

5.2 Applications
The key application of this work is to develop an attention-aware version of Guru that detects and combats MW in real-time. Once the goal of MW detection is realized, Guru has a number of paths to pursue to re-engage attention.

At an immediate level, one initial effect of MW is that the student simply fails to attend to a unit of information or a salient event in the learning environment. The unattended information, question, or event is needed to construct an adequate mental model so that subsequent knowledge can be assimilated or the student will be left behind. Thus, a simple direct approach is to reassert the missed information (“e.g., Mary, let me repeat that....”) or highlight the information by directing attention to specific areas of the display (e.g., “Mary, you might want to look at the highlighted image showing the chromosomes duplicating”). Taking a somewhat different approach, Guru can also launch a sub-dialogue where it asks a content-specific question (e.g., “Mary, what happens to the chromosomes when they duplicate?”) or asks the student to complete a mini-activity (e.g., “Mary, we now have a simulation of the first phase in mitosis. Can you....”). Guru can also ask the student to self-explain when MW is detected.

Additional measures might be needed if MW persists despite these intervention strategies. One option is to simply change to a new activity. Guru might even suggest changing topics or offering a choice for what students would like to do next. If all else fails, Guru might even suggest that the student take a break.

It is important to note that the proposed intervention strategies rely on MW detection, which is inherently imperfect. The detector might inaccurately assert that a student is MW when they are not (false alarms) or it might assert that a student is actively attending when they are in fact MW (misses). MW detection does not need to be perfect as long as we account for this in MW interventions. For example, Guru can adopt a probabilistic approach where the MW detector provides an estimate of the likelihood that the student is MW. This likelihood will guide whether an intervention is launched (i.e. if the likelihood of MW is 70%, there is a 70% chance that an intervention will be triggered). Second, interventions can be designed to be “fail-soft” in that there are no harmful effects if delivered incorrectly.

5.3 Limitations and Future Work
There were several limitations with this study. One key limitation pertains to the moderate MW detection accuracy. Although, we detected MW above chance levels using several different classifiers, these results leave room for improvement. Ongoing work seeks to reduce the false positive rate while increasing the hit rate for our MW models by expanding our feature set and incorporating temporal information in the machine learning.

We designed our approach to include a low-cost eye tracker, however, these consumer models have a lower sampling-rate, limiting the accuracy of the eye-gaze data compared to research-grade eye trackers. Furthermore, although we desire to eventually deploy our system in noisy classroom environments, we only tested our system in a quiet lab setting.

This work is also limited by the features used in the supervised learning process, which were a small and potentially restrictive subset of gaze features. We also did not model temporal patterns of eye movements, such as examining if the participant revisited an area of the screen they had previously viewed. Additionally, we only used a small number of contextual features. Future work may consider utilizing log files from the tutoring session more extensively to create more in-depth context features (e.g., content, timing, and length of student responses, etc.).

The results of this study invite several avenues for improvement which we will explore as future work. First, we will explore additional eye-gaze features, such as those that track localized regions of interest but at a level of abstraction that does not limit generalizability to additional interfaces. Informed by our observation that the inclusion of contextual features improved detection of MW, we will explore additional contextual features from the ITS, again with an eye for more generalizable features (e.g., response time). Furthermore, it is possible to build multiple MW detector specialized for different phases in the Guru tutoring sessions, although this would require a large amount of data and would make these detectors less able to generalize to other ITSs. Finally, we will collect data in the real-world context of a computer-enabled classroom where 20-30 students interact with Guru on individual computers while their gaze is being tracked. Indeed, preliminary data collection on this front is already underway.

5.4 Concluding Remarks
Attention is a crucial part of learning. An attention-aware ITS that can detect a student’s attentional state as well as redirect their attention to better engage them in the learning task could be very beneficial for engagement and learning. Attention-awareness, however, requires monitoring of attention, which has historically been limited to the lab. However, advances in consumer-grade eye-tracking have opened up the possibility of gaze tracking during learning with ITSs and other technologies, thereby enabling a new generation of attention-aware cyberlearning.

6. ACKNOWLEDGMENTS
This research was supported by the National Science Foundation (NSF) (DRL 1235958 and IIS 1523091). Any opinions, findings and conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF.

7. REFERENCES


How Deep is Knowledge Tracing?

Mohammad Khajah
Dept. of Computer Science
University of Colorado
Boulder, Colorado 80309
mohammad.khajah@colorado.edu

Robert V. Lindsey
Dept. of Computer Science
University of Colorado
Boulder, Colorado 80309
robert.lindsey@colorado.edu

Michael C. Mozer
Dept. of Computer Science
University of Colorado
Boulder, Colorado 80309
mozer@colorado.edu

ABSTRACT

In theoretical cognitive science, there is a tension between highly structured models whose parameters have a direct psychological interpretation and highly complex, general-purpose models whose parameters and representations are difficult to interpret. The former typically provide more insight into cognition but the latter often perform better. This tension has recently surfaced in the realm of educational data mining, where a deep learning approach to predicting students' performance as they work through a series of exercises—termed deep knowledge tracing or DKT—has demonstrated a stunning performance advantage over the mainstay of the field, Bayesian knowledge tracing or BKT. In this article, we attempt to understand the basis for DKT's advantage by considering the sources of statistical regularity in the data that DKT can leverage but which BKT cannot. We hypothesize four forms of regularity that BKT fails to exploit: recency effects, the contextualized trial sequence, inter-skill similarity, and individual variation in ability. We demonstrate that when BKT is extended to allow it more flexibility in modeling statistical regularities—using extensions previously proposed in the literature—BKT achieves a level of performance indistinguishable from that of DKT. We argue that while DKT is a powerful, useful, general-purpose framework for modeling student learning, its gains do not come from the discovery of novel representations—the fundamental advantage of deep learning. To answer the question posed in our title, knowledge tracing may be a domain that does not require 'depth'; shallow models like BKT can perform just as well and offer us greater interpretability and explanatory power.

1. INTRODUCTION

In the past forty years, machine learning and cognitive science have undergone many paradigm shifts, but few have been as dramatic as the recent surge of interest in deep learning [16]. Although deep learning is little more than a re-branding of neural network techniques popular around 1990, deep learning has achieved some remarkable results thanks to much faster computing resources and much larger data sets than were available in 1990. Deep learning underlies state-of-the-art systems in speech recognition, language processing, and image classification [16, 26]. Deep learning also is responsible for systems that can produce captions for images [29], create synthetic images [9], play video games [19] and even Go [27].

The ‘deep’ in deep learning refers to multiple levels of representation transformation that lie between model inputs and outputs. For example, an image-classification model may take pixel values as input and produce a labeling of the objects in the image as output. Between the input and output is a series of representation transformations that construct successively higher-order features—features that are less sensitive to lighting conditions and the position of objects in the image, and more sensitive to the identities of the objects and their qualitative relationships. The features discovered by deep learning exhibit a complexity and subtlety that make them difficult to analyze and understand (e.g., [31]). Furthermore, no human engineer could wire up a solution as thorough and accurate as solutions discovered by deep learning. Deep learning models are fundamentally non-parametric, in the sense that interpreting individual weights and individual unit activations in a network is pretty much impossible. This opacity is in stark contrast to parametric models, e.g., linear regression, where each of the coefficients has a clear interpretation in terms of the problem at hand and the input features.

In one domain after the next, deep learning has achieved gains over traditional approaches. Deep learning discards hand-crafted features in favor of representation learning, and deep learning often ignores domain knowledge and structure in favor of massive data sets and general architectural constraints on models (e.g., models with spatial locality to process images, and models with local temporal constraints to process time series).

It was inevitable that deep learning would be applied to student-learning data [22]. This domain has traditionally been the purview of the educational data mining community, where Bayesian knowledge tracing, or BKT, is the dominant computational approach [3]. The deep learning approach to modeling student data, termed deep knowledge tracing or DKT, created a buzz when it appeared at the Neural Information Processing Systems Conference in December 2015, including press inquiries (N. Heffernan, personal commun—
In this article, we explore the success of DKT. One approach to this exploration might be to experiment with DKT, removing components of the model or modifying the input data to determine which model components and data characteristics are essential to DKT’s performance. We pursue an alternative approach in which we first formulate hypotheses concerning the signals in the data that DKT is able to exploit but that BKT is not. Given these hypotheses, we propose extensions to BKT which provide it with additional flexibility, and we evaluate whether the enhanced BKT can achieve results comparable to DKT. This procedure leads not only to a better understanding of how BKT and DKT differ, but also helps us to understand the structure and statistical regularities in the data source.

1.1 Modeling Student Learning

The domain we’re concerned with is electronic tutoring systems which employ cognitive models to track and assess student knowledge. Beliefs about what a student knows and doesn’t know allow a tutoring system to dynamically adapt its feedback and instruction to optimize the depth and efficiency of learning.

Ultimately, the measure of learning is how well students are able to apply skills that they have been taught. Consequently, student modeling is often formulated as time series prediction: given the series of exercises a student has attempted previously and the student’s success or failure on each exercise, predict how the student will fare on a new exercise. Formally, the data consist of a set of binary random variables indicating whether student $s$ produces a correct response on trial $t$. $\{X_{st}\}$. The data also include the exercise labels, $\{Y_{st}\}$, which characterize the exercise. Secondary data has also been incorporated in models, including the student’s utilization of hints, response time, and characteristics of the specific exercise and the student’s particular history with related exercises [2, 30]. Although such data improve predictions, the bulk of research in this area has focused on the primary measure—whether a response is correct or incorrect—and a sensible research strategy is to determine the best model based on the primary data, and then to determine how to incorporate secondary data.

The exercise label, $Y_{st}$, might index the specific exercise, e.g., $3 + 4$ versus $2 + 6$, or it might provide a more general characterization of the exercise, e.g., single digit addition. In the latter case, exercise are grouped by the skill that must be applied to obtain a solution. Although we will use the term skill in this article, others refer to the skill as a knowledge component, and the authors of DKT also use the term concept. Regardless, the important distinction for the purpose of our work is between a label that indicates the particular exercise and a label that indicates the general skill required to perform the exercise. We refer to these two types of labels as exercise indexed and skill indexed, respectively.

1.2 Knowledge Tracing

DKT models skill-specific performance, i.e., performance on a series of exercises that all tap the same skill. A separate instantiation of BKT is made for each skill, and a student’s raw trial sequence is parsed into skill-specific subsequences that preserve the relative ordering of exercises within a skill but discard the ordering relationship of exercises across skills. For a given skill $\sigma$, BKT is trained using the data from each student $s$, $\{X_{si}|Y_{si} = \sigma\}$, where the relative trial order is preserved. Because it will become important for us to distinguish between absolute trial index and the relative trial index within a skill, we use $t$ to denote the former and use $i$ to denote the latter.

DKT is based on a theory of all-or-none human learning [1] which postulates that the knowledge state of student $s$ following the $i$th exercise requiring a certain skill, $K_{si}$, is binary: 1 if the skill has been mastered, 0 otherwise. BKT, formalized as a hidden Markov model, infers $K_{si}$ from the sequence of observed responses on trials $1...i$, $\{X_{s1}, X_{s2},... X_{si}\}$. BKT is typically specified by four parameters: $P(K_{si} = 1)$, the probability that the student has mastered the skill prior to solving the first exercise; $P(K_{si+1} = 1 | K_{si} = 0)$, the transition probability from the not-mastered to mastered state; $P(X_{si} = 1 | K_{si} = 0)$, the probability of correctly guessing the answer prior to skill mastery; and $P(X_{si} = 0 | K_{si} = 1)$, the probability of answering incorrectly due to a slip following skill mastery. Because BKT is typically used in modeling practice over brief intervals, the model assumes no forgetting, i.e., $K$ cannot transition from 1 to 0.

BKT is a highly constrained, structured model. It assumes that the student’s knowledge state is binary, that predicting performance on an exercise requiring a given skill depends only on the student’s binary knowledge state, and that the skill associated with each exercise is known in advance. If correct, these assumptions allow the model to make strong inferences. If incorrect, they limit the model’s performance. The only way to determine if model assumptions are correct is to construct an alternative model that makes different assumptions and to determine whether the alternative outperforms BKT. DKT is exactly this alternative model, and its strong performance directs us to examine BKT’s limitations. First, however, we briefly describe DKT.

Rather than constructing a separate model for each skill, DKT models all skills jointly. The input to the model is the complete sequence of exercise-performance pairs, $\{(X_{s1}, Y_{s1})... (X_{sT-1}, Y_{sT-1}) (X_{sT}, Y_{sT})\}$, presented one trial at a time. As depicted in Figure 1, DKT is a recurrent neural net which takes $(X_{si}, Y_{si})$ as input and predicts $X_{si+1}$ for each possible exercise label. The model is trained and evaluated based on the match between the actual and predicted $X_{si+1}$ for the tested exercise $(Y_{si+1})$. In addition to the input and output layers representing the current trial and the next trial, respectively, the network has a hidden layer with fully recurrent connections (i.e., each hidden unit connects back to all other hidden units). The hidden layer thus serves to retain relevant aspects of the input history as they are use-
Figure 1: Deep knowledge tracing (DKT) architecture. Each rectangle depicts a set of processing units; each arrow depicts complete connectivity between each unit in the source layer and each unit in the destination layer.

1.3 Where Does BKT Fall Short?

In this section, we describe four regularities that we conjecture to be present in the student-performance data. DKT is flexible enough that it has the potential to discover these regularities, but the more constrained BKT model is simply not crafted to exploit the regularities. In following sections, we suggest means of extending BKT to exploit such regularities, and conduct simulation studies to determine whether the enhanced BKT achieves performance comparable to that of DKT.

1.3.1 Recency Effects

Human behavior is strongly recency driven. For example, when individuals perform a choice task repeatedly, response latency can be predicted by an exponentially decaying average of recent stimuli [12]. Intuitively, one might expect to observe recency effects in student performance. Consider, for example, a student’s time varying engagement. If the level of engagement varies slowly relative to the rate at which exercises are being solved, a correlation would be induced in performance across local spans of time. A student who performed poorly on the last trial because they were distracted is likely to perform poorly on the current trial. We conducted a simple assessment of recency using the Assistments data set (the details of this data set will be described shortly). Similarly to [5], we built an autoregressive model that predicts performance on the current trial as an exponentially weighted average of performance on past trials, with a decay half life of about 5 steps. We found that this single parameter model fit the Assistments data reliably better than classic BKT. (We are not presenting details of this simulation because we will evaluate a more rigorous variant of the idea in a following section. Our goal here is to convince the reader that there is likely some value to the notion of recency-weighted prediction.)

Recurrent neural networks tend to be more strongly influenced by recent events in a sequence than more distant events [20]. Consequently, DKT is well suited to exploiting recent performance in making predictions. In contrast, the generative model underlying BKT supposes that once a skill is learned, performance will remain strong, and that a slip at time \( t \) is independent of a slip at \( t + 1 \).

1.3.2 Contextualized Trial Sequence

The psychological literature on practice of multiple skills indicates that the sequence in which an exercise is embedded influences learning and retention (e.g., [24, 25]). For example, given three exercises each of skills \( A \) and \( B \), presenting the exercises in the interleaved order \( A_1-B_1-A_2-B_2-A_3-B_3 \) yields superior performance relative to presenting the exercises in the blocked order \( A_1-A_2-A_3-B_1-B_2-B_3 \). (Performance in this situation can be based on an immediate or delayed test.)

Because DKT is fed the entire sequence of exercises a student receives in the order the student receives them, it can potentially infer the effect of exercise order on learning. In contrast, because classic BKT separates exercises by skill, preserving only the relative order of exercises within a skill, the training sequence for BKT is the same regardless of whether the trial order is blocked or interleaved.

1.3.3 Inter-Skill Similarity

Each exercise presented to a student has an associated label. In typical applications of BKT—as well as two of the three simulations reported in Piech et al. [22]—the label indicates the skill required to solve the problem. Any two such skills, \( S_1 \) and \( S_2 \), may vary in their degree of relatedness. The stronger the relatedness, the more highly correlated one would expect performance to be on exercises tapping the two skills, and the more likely that the two skills will be learned simultaneously.

DKT has the capacity to encode inter-skill similarity. If each hidden unit represents student knowledge state for a particular skill, then the hidden-to-hidden connections encode the degree of overlap. In an extreme case, if two skills are highly similar, they can be modeled by a single hidden knowledge state. In contrast, classic BKT treats each skill as an in-
dependent modeling problem and thus can not discover or leverage inter-skill similarity.

DKT has the additional strength, as demonstrated by Piech et al., that it can accommodate the absence of skill labels. If each label simply indexes a specific exercise, DKT can discover interdependence between exercises in exactly the same manner as it discovers interdependence between skills. In contrast, BKT requires exercise labels to be skill indexed.

1.3.4 Individual Variation in Ability

Students vary in ability, as reflected in individual differences in mean accuracy across trials and skills. Individual variation might potentially be used in a predictive manner: a student’s accuracy on early trials in a sequence might predict accuracy on later trials, regardless of the skills required to solve exercises. We performed a simple verification of this hypothesis using the ASSISTMENTS data set. In this data set, students study one skill at a time and then move on to the next skill. We computed correlation between mean accuracy of all trials on the first n skills and the mean accuracy of all trials on skill n + 1, for all students and for n ∈ {1,...,N − 1} where N is the number of skills a student studied. We obtained a correlation coefficient of 0.39: students who tend to do well on the early skills learned tend to do well on later skills, regardless of the skills involved.

DKT is presented with a student’s complete trial sequence. It can use a student’s average accuracy up to trial t to predict trial t + 1. Because BKT models each skill separately from the others, it does not have the contextual information needed to estimate a student’s average accuracy or overall ability.

2. EXTENDING BKT

In the previous section, we described four regularities that appear to be present in the data and which we conjecture that DKT exploits but which the classic BKT model cannot. In this section, we describe three extensions to BKT that would bring BKT on par with DKT with regard to these regularities.

2.1 Forgetting

To better capture recency effects, BKT can be augmented to allow for forgetting of skills. Forgetting corresponds to fitting a BKT parameter $F \equiv P(K_s,i+1 = 0 \mid K_s = 1)$, the probability of transitioning from a state of knowing to not knowing a skill. In standard BKT, $F = 0$.

Without forgetting, once BKT infers that the student has learned, even a long run of poorly performing trials cannot alter the inferred knowledge state. However, with forgetting, the knowledge state can transition in either direction, which allows the model to be more sensitive to the recent trials: A run of unsuccessful trials is indicative of not knowing the skill, regardless of what preceded the run. Forgetting is not a new idea to BKT, and in fact was included in the original psychological theory that underlies the notion of binary knowledge state [1]. However, it has not typically been incorporated into BKT. When it has been included in BKT [22], the motivation was to model forgetting from one day to the next, not forgetting that can occur on a much shorter time scale.

Incorporating forgetting can not only sensitize BKT to recent events but can also contextualize trial sequences. To explain, consider an exercise sequence such as $A_1-A_2-B_1-A_3-B_2-B_3-A_4$, where the labels are instances of skills $A$ and $B$. Ordinary BKT discards the absolute number of trials between two exercises of a given skill, but with forgetting, we can count the number of intervening trials and treat each as an independent opportunity for forgetting to occur. Consequently, the probability of forgetting between $A_1$ and $A_2$ is $F$, but the probability of forgetting between $A_2$ and $A_3$ is $1 - (1 - F)^2$ and between $A_3$ and $A_4$ is $1 - (1 - F)^3$. Using forgetting, BKT can readily incorporate some information about the absolute trial sequence, and therefore has more potential than classic BKT to be sensitive to interspersed trials in the exercise sequence.

2.2 Skill Discovery

To model interactions among skills, one might suppose that each skill has some degree of influence on the learning of other skills, not unlike the connection among hidden units in DKT. For BKT to allow for such interactions among skills, the independent BKT models would need to be interconnected, using an architecture such as a factorial hidden Markov model [6]. As an alternative to this somewhat complex approach, we explored a simpler scheme in which different exercise labels could be collapsed together to form a single skill. For example, consider an exercise sequence such as $A_1-B_1-A_2-C_1-B_2-C_2-C_3$. If skills $A$ and $B$ are highly similar or overlapping, such that learning one predicts learning the other, it would be more sensible to treat this sequence in a manner that groups $A$ and $B$ into a single skill, and to train a single BKT instantiation on both $A$ and $B$ trials. This approach can be used whether the exercise labels are skill indices or exercise indices. (One of the data sets used by Piech et al. [22] to motivate DKT has exercise-indexed labels).

We recently proposed an inference procedure that automatically discovers the cognitive skills needed to accurately model a given data set [18]. (A related procedure was independently proposed in [8].) The approach couples BKT with a technique that searches over partitions of the exercise labels to simultaneously (1) determine which skill is required to correctly answer each exercise, and (2) model a student’s dynamical knowledge state for each skill. Formally, the technique assigns each exercise label to a latent skill such that a student’s expected accuracy on a sequence of time-skill exercises improves monotonically with practice according to BKT. Rather than discarding the skills identified by experts, our technique incorporates a nonparametric prior over the exercise-skill assignments that is based on the expert-provided skills and a weighted Chinese restaurant process [11].

In the above illustration, our technique would group $A$ and $B$ into one skill and $C$ into another. This procedure collapses like skills (or like exercises), yielding better fits to the data by BKT. Thus, the procedure performs a sort of skill discovery.

2.3 Incorporating Latent Student-Abilities

To account for individual variation in student ability, we have extended BKT [14, 13] such that slip and guess prob-
abilities are modulated by a latent ability parameter that is inferred from the data, much in the spirit of item-response theory [4]. As we did in [14], we assume that students with stronger abilities have lower slip and higher guess probabilities. When the model is presented with new students, the posterior predictive distribution on abilities is used initially, but as responses from the new student are observed, uncertainty in the student’s ability diminishes, yielding better predictions for the student.

3. SIMULATIONS

3.1 Data Sets

Piech et al. [22] studied three data sets. One of the data sets, from Khan Academy, is not publicly available. Despite our requests and a plea from one of the co-authors of the DKT paper, we were unable to obtain permission from the data science team at Khan Academy to use the data set. We did investigate the other two data sets in Piech et al., which are as follows.

ASSISTMENTS is an electronic tutor that teaches and evaluates students in grade-school math. The 2009-2010 “skill builder” data set is a large, standard benchmark, available by searching the web for assistment-2009-2010-data. We used the train/test split provided by Piech et al., and following Piech et al., we discarded all students who had only a single trial of data.

SYNTHETIC is a synthetic data set created by Piech et al. to model virtual students learning virtual skills. The training and test sets each consist of 2000 virtual students performing the same sequence of 50 exercises drawn from 5 skills. The exercise on trial \( t \) is assumed to have a difficulty characterized by \( \delta_t \) and require a skill specified by \( \sigma_t \). The exercises are labeled by the identity of the exercise, not by the underlying skill, \( \sigma_t \). The ability of a student, denoted, \( \alpha_t \), varies over time according to a drift-diffusion process, generally increasing with practice. The response correctness on trial \( t \) is a Bernoulli draw with probability specified by guessing-corrected item-response theory with difficulty and ability parameters \( \delta_t \) and \( \alpha_t \). This data set is challenging for BKT because the skill assignments, \( \sigma_t \), are not provided and must be inferred from the data. Without the skill assignments, BKT must be used either with all exercises associated with a single skill or each exercise associated with its own skill. Either of these assumptions will miss important structure in the data. SYNTHETIC is an interesting data set in that the underlying generative model is neither a perfect match to DKT or BKT (even with the enhancements we have described). The generative model seems realistic in its assumption that knowledge state varies continuously.

We included two additional data sets in our simulations. SPANISH is a data set of 182 middle-school students practicing 409 Spanish exercises (translations and application of simple skills such as verb conjugation) over the course of a 15-week semester, with a total of 578,726 trials [17]. STATICS is from a college-level engineering statics course with 189,297 trials and 333 students and 1,223 exercises [28], available from the PSLC DataShop web site [15].

3.2 Methods

We evaluated five variants of BKT, each of which incorporates a different subset of the extensions described in the previous section: a base version that corresponds to the classic model and the model against which DKT was evaluated in [22], which we’ll refer to simply as BKT; a version that incorporates forgetting (BKT+F), a version that incorporates skill discovery (BKT+S), a version that incorporates latent abilities (BKT+A), and a version that incorporates all three of the extensions (BKT+FSA). We also built our own implementation of DKT with LSTM recurrent units (Piech et al. described the LSTM version as better performing, but posted only the code for the standard recurrent neural net version.) We verified that our implementation produced results comparable to those reported in [22] on ASSISTMENTS and SYNTHETIC. We then also ran the model on SPANISH and STATICS.

For ASSISTMENTS, SPANISH, and STATICS, we used a single train/test split. The ASSISTMENTS train/test split was identical to that used by Piech et al. For SYNTHETIC, we used the 20 simulation sets provided by Piech et al. and averaged results across the 20 simulations.

Each model was evaluated on each domain’s test data set, and the performance of the model was quantified with a discriminability score, the area under the ROC curve or AUC. AUC is a measure ranging from .5, reflecting no ability to discriminate correct from incorrect responses, to 1.0, reflecting perfect discrimination. AUC is computed by obtaining a prediction on the test set for each trial, across all skills, and then using the complete set of predictions to form the ROC curve. Although Piech et al. [22] do not describe the procedure they use to compute AUC for DKT, code they have made available implements the procedure we describe, and not the obvious alternative procedure in which ROC curves are computed on a per-skill basis and then averaged to obtain an overall AUC.

3.3 Results

Figure 2 presents the results of our comparison of five variants of BKT on the four data sets. We walk through the data sets from left to right.

On ASSISTMENTS, classic BKT obtains an AUC of 0.73, better than the 0.67 reported for BKT by Piech et al. We are not sure why the scores do not match, although 0.67 is close to the AUC score we obtain if we treat all exercises as associated with a single skill or if we compute AUC on a per-skill basis and then average.\(^1\) BKT+F obtains an AUC of 0.85, BKT+A

\(^1\)https://github.com/robert-lindsey/WCRP/tree/forgetting

\(^2\)https://github.com/mmkhajah/dkt

\(^3\)Piech et al. cite Pardos and Heffernan [21] as obtaining BKT’s best reported performance on ASSISTMENTS—an AUC of 0.69. In [21], the overall AUC is computed by averaging the per-skill AUCs. This method yields a lower score than the method used by Piech et al., for two reasons. First, the Piech procedure weighs all trials equally, whereas the Pardos and Heffernan procedure weighs all skills equally. With the latter procedure, the overall AUC will be dinged if the model does poorly on a skill with just a few trials, as we have observed to be the case with ASSISTMENTS. The latter procedure also produces a lower overall AUC because it suppresses any lift due to being able to predict the relative accuracy of different skills. In summary, it appears that
not quite as good as the 0.86 value reported for DKT by Piech et al. Examining the various enhancements to BKT, AUC is boosted both by incorporating forgetting and by incorporating latent student abilities. We find it somewhat puzzling that the combination of the two enhancements, embodied in BKT+FSA, does no better than BKT+F or BKT+A, considering that the two enhancements tap different properties of the data: the student abilities help predict transfer from one skill to the next, whereas forgetting facilitates prediction within a skill.

To summarize the comparison of BKT and DKT, 31.6% of difference in performance reported in [22] appears to be due to the use of a biased procedure for computing the AUC for BTK. Another 50.6% of the difference in performance reported vanishes if BKT is augmented to allow for forgetting. We can further improve BKT if we allow the skill discovery algorithm to operate with exercise labels that index individual exercises, as opposed to labels that index the skill associated with each exercise. With exercise-indexed labels, BKT+S and BKT+FSA both obtain an AUC of 0.90, beating DKT. However, given DKT’s ability to perform skill discovery, we would not be surprised if it also achieved a similar level of performance when allowed to exploit exercise-indexed labels.

Turning to Synthetic, classic BKT obtains an AUC of 0.62, again significantly better than the 0.54 reported by Piech et al. In our simulation, we treat each exercise as having a distinct skill label, and thus BKT learns nothing more than the mean performance level for a specific exercise. (Because the exercises are presented in a fixed order, the exercise identity and the trial number are confounded. Because performance tends to improve as trials advance in the synthetic data, BKT is able to learn this relationship.) It is possible here that Piech et al. treated all exercises as associated with a single skill or that they used the biased procedure for computing AUC; either of these explanations is consistent with their reported AUC of 0.54.

Regarding the enhancements to BKT, adding student abilities (BKT+A) improves prediction of Synthetic which is understandable given that the generative process simulates students with abilities that vary slowly over time. Adding forgetting (BKT+F) does not help, consistent with the generative process which assumes that knowledge level is on average increasing with practice; there is no systematic forgetting in the student simulation. Critical to this simulation is skill induction: BKT+S and BKT+FSA achieve an AUC of 0.80, better than the reported 0.75 for DKT in [22].

On Statics, each BKT extension obtains an improvement over classic BKT, although the magnitude of the improvements are small. The full model, BKT+FSA, obtains an AUC of 0.75 and our implementation of DKT obtains a nearly identical AUC of 0.76. On Spanish, the BKT extensions obtain very little benefit. The full model, BKT+FSA, obtains an AUC of 0.846 and again, DKT obtains a nearly identical AUC of 0.836. These two sets of results indicate that for at least some data sets, classic BKT has no glaring deficiencies. However, we note that BKT model accuracy can be improved if algorithms are considered that use exercise labels which are indexed by exercise and not by skill. For example, with Statics, performing skill discovery using exercise-indexed labels, [17] obtain an AUC of 0.81, much better than the score of 0.73 we report here for BKT+S based on skill-indexed labels.

In summary, enhanced BKT appears to perform as well on average as DKT across the four data sets. Enhanced BKT outperforms DKT by 20.0% (.05 AUC units) on Synthetic and by 3.6% (.01 AUC unit) on Statics. Enhanced BKT underperforms DKT by 3.3% (.03 AUC units) on Assistments and by 3.5% (.01 AUC unit) on Spanish. These percentages are based on the difference of AUCs scaled by 0.5, which takes into account the fact that an AUC of 0.5 indicates no discriminability.

Proceedings of the 9th International Conference on Educational Data Mining
4. DISCUSSION

Our goal in this article was to investigate the basis for the impressive predictive advantage of deep knowledge tracing over Bayesian knowledge tracing. We found some evidence that different procedures may have been used to evaluate DKT and BKT in [22], leading to a bias against BKT. When we replicated simulations of BKT reported in [22], we obtained significantly better performance: an AUC of 0.73 versus 0.67 on ASSISTMENTS, and an AUC of 0.62 versus 0.54 on SYNTHETIC.

However, even when the bias is eliminated, DKT obtains real performance gains over BKT. To understand the basis for these gains, we hypothesized various forms of regularity in the data which BKT is not able to exploit. We proposed enhancements to BKT to allow it to exploit these regularities, and we found that the enhanced BKT achieved a level of performance on average indistinguishable from that of DKT over the four data sets tested. The enhancements we explored are not novel; they have previously been proposed and evaluated in the literature. They include forgetting [23], latent student abilities [14, 13, 21], and skill induction [17, 8].

We observe that different enhancements to BKT matter for different data sets. For ASSISTMENTS, incorporating forgetting is key; forgetting allows BKT to capture recency effects. For SYNTHETIC, incorporating skill discovery yielded huge gains, as one would expect when the exercise-skill mapping is not known. And for STATICS, incorporating latent student abilities was relatively most beneficial; these abilities enable the model to tease apart the capability of a student and the intrinsic difficulty of an exercise or skill. Of the three enhancements, forgetting and student abilities are computationally inexpensive to implement, whereas skill discovery adds an extra layer of computational complexity to inference.

The elegance of DKT is apparent when one considers the effort we have invested to bring BKT to par with DKT. DKT did not require its creators to analyze the domain and determine sources of structure in the data. In contrast, our approach to augmenting BKT required some domain expertise, a thoughtful analysis of BKT’s limitations, and distinct solutions to each limitation. DKT is a generic recurrent neural network model [10], and it has no constructs that are specialized to modeling learning and forgetting, discovering skills, or inferring student abilities. This flexibility makes DKT robust on a variety of datasets with little prior analysis of the domains. Although training recurrent networks is computationally intensive, tools exist to exploit the parallel processing power in graphics processing units (GPUs), which means that DKT can scale to large datasets. Classic BKT is inexpensive to fit, although the variants we evaluated—particularly the model that incorporates skill discovery—require computation-intensive MCMC methods that have a distinct set of issues when it comes to parallelization.

DTK’s advantages come at a price: interpretability. DKT is massive neural network model with tens of thousands of parameters which are near-impossible to interpret. Although the creators of DKT did not have to invest much up-front time analyzing their domain, they did have to invest substantive effort to understand what the model had actually learned. Our proposed BKT extensions achieve predictive performance similar to DKT whilst remaining interpretable: the model parameters (forgetting rate, student ability, etc.) are psychologically meaningful. When skill discovery is incorporated into BKT, the result is clear: a partition of exercises into skills. Reading out such a partitioning from DKT is challenging and only an approximate representation of the knowledge in DKT.

Finally, we return to the question posed in the paper’s title: How deep is knowledge tracing? Deep learning refers to the discovery of representations. Our results suggest that representation discovery is not at the core of DKT’s success. We base this argument on the fact that our enhancements to BKT bring it to the performance level of DKT without requiring any sort of subsymbolic representation discovery. Representation discovery is clearly critical in perceptual domains such as image or speech classification. But the domain of education and student learning is high level and abstract. The input and output elements of models are psychologically meaningful. The relevant internal states of the learner have some psychological basis. The characterization of exercises and skills can—to at least a partial extent—be expressed symbolically.

Instead of attributing DKT’s success to representation discovery, we attribute DKT’s success to its flexibility and generality in capturing statistical regularities directly present in the inputs and outputs. As long as there are sufficient data to constrain the model, DKT is more powerful than classic BKT. BKT arose in a simpler era, an era in which data and computation resources were precious. DKT reveals the value of relaxing these constraints in the big data era. But despite the wild popularity of deep learning, there are many ways to relax the constraints and build more powerful models other than creating a black box predictive device with a vast interconnected tangle of connections and parameters that are nearly impossible to interpret.

5. ACKNOWLEDGMENTS

This research was supported by NSF grants SES-1461535, SBE-0542013, and SMA-1041755.

6. REFERENCES


4Of course, the skill discovery mechanism we incorporated certainly does regroup exercises to form skills, but the form of this regrouping or partitioning is far more limited than the typical transformations in a neural network to map from one level of representation to another.
Temporally Coherent Clustering of Student Data

Severin Klingler  
Department of Computer Science  
ETH Zurich, Switzerland  
kseverin@inf.ethz.ch

Tanja Käser  
Department of Computer Science  
ETH Zurich, Switzerland  
kaesert@inf.ethz.ch

Barbara Solenthaler  
Department of Computer Science  
ETH Zurich, Switzerland  
sobarbar@inf.ethz.ch

Markus Gross  
Department of Computer Science  
ETH Zurich, Switzerland  
grossm@inf.ethz.ch

ABSTRACT
The extraction of student behavior is an important task in educational data mining. A common approach to detect similar behavior patterns is to cluster sequential data. Standard approaches identify clusters at each time step separately and typically show low performance for data that inherently suffer from noise, resulting in temporally inconsistent clusters. We propose an evolutionary clustering pipeline that can be applied to learning data, aiming at improving cluster stability over multiple training sessions in the presence of noise. Our model selection is designed such that relevant cluster evolution effects can be captured. The pipeline can be used as a black box for any intelligent tutoring system (ITS). We show that our method outperforms previous work regarding clustering performance and stability on synthetic data. Using log data from two ITS, we demonstrate that the proposed pipeline is able to detect interesting student behavior and properties of learning environments.

Keywords
Evolutionary Clustering, Markov Chains, Sequence Mining, Distance Metrics

1. INTRODUCTION
The extraction of student properties is a central element in educational data mining. On the one hand, the identification of student abilities and behavior patterns allows us to draw conclusions about human learning. On the other hand, the extracted properties can be used to improve the adaptation of the underlying intelligent tutoring system (ITS).

Clustering of sequential data is a common approach to detect similar behavior patterns and has been successfully applied to a variety of applications such as reading comprehension [22], online collaboration tools [24], table-top environments [19], web browsing [25], physics simulations [4] or homework assignments [11]. Furthermore, a variety of different student behavior has been investigated, [20] identified students that impose challenges for the student models. Other work studied the relation between interaction patterns and the performance of students [3, 14] and the relation between student action sequences and their affective states [3].

Common techniques for the analysis of sequential data include sequence mining [1, 19], differential pattern mining [11] or Hidden Markov models (HMM) [5, 6]. Sequential pattern mining techniques have been contextualized using piece-wise linear segmentation [14]. Others have employed semi-supervised graph clustering using the predictions from a student model as additional constraints [20]. Clustering sequential data employing similarity measures on state sequences was used in [4, 8]. These state sequences can be aggregated into Markov Chains modeling the state transitions [17]. HMM have been employed to extract stable groups from temporal data by joint optimization of the model parameters and the cluster count [18].

While the previous work discussed above analyze student clusters at a given point in time, a temporal analysis would allow to identify how interaction patterns change over time and how groups of similar students evolve. Temporal effects of cluster evolution have been analyzed in [15], based on static clustering at each time step. Static approaches are sensitive to noise in the data and may result in temporally inconsistent clusters. Evolutionary clustering methods [7] address this problem as they consider multiple subsequent time steps. The temporal smoothing increases the resulting cluster stability notably and allows for a better analysis of the clusters, i.e., the student properties and interaction patterns. Recently, an evolutionary clustering approach called AFFECT [27] has been introduced that smooths proximities of students over time followed by static clustering. AFFECT was shown to outperform static clustering algorithms.

In this paper, we present a complete processing pipeline for evolutionary clustering that can be used as a black box for any ITS. We incorporate a variation of the AFFECT method into our pipeline and demonstrate that temporal smoothing has beneficial properties for extracting student behavior and groups from educational data. We propose several extensions of the original method tailored towards learning data. Our approach is articulated in four steps. In a first step, we extract action sequences from ITS log data and aggregate them using Markov Chains. We show that the Markov Chain representation of the actions is superior to direct sequence mining techniques [4, 17] with respect to noise cancellation and the ability to identify groups of students with similar behavior. The second step consists of computing pairwise similarities between the Markov Chains. While the proposed pipeline provides flexibility in the choice of similarity measure, the Hellinger distance outperforms other metrics that

Proceedings of the 9th International Conference on Educational Data Mining 102
are frequently used in the educational data mining literature \cite{4,17}. Based on the obtained similarities, evolutionary clustering \cite{27} is performed in the third step. The temporal aspect of the student data leads to changing behavior patterns, i.e., we expect the number of clusters and cluster sizes to change over time. Therefore, capturing cluster evolution events, such as merging, splitting, dissolving and forming of clusters, is crucial in order to analyze sequential data. To capture these events automatically, we compute the optimal cluster count for each time step using the AICc criterion.

Using synthetic data, we demonstrate that our method exhibits a higher performance and is more robust to noise than previous work \cite{4,17}. We further show that our pipeline is able to extract stable clusters over time and reliably detects all cluster events. In an exploratory analysis on real-world data, we apply our pipeline to log data from two different ITSs: One for spelling learning and one for mathematics learning. Finally, we present a set of visual tools that are powerful to analyze temporal data and student clusters.

2. METHOD

Our method for student clustering is designed to address two challenges when clustering temporal data. First, the method provides temporally consistent clusters. Second, our pipeline is able to capture changes in cluster sizes as well as in the number of clusters. Four cluster events are of particular interest in the context of educational data mining: merging, splitting, dissolving and forming of clusters. If the behavior of students from two different clusters becomes more similar over time, we expect the clusters to merge (this could mark a training effect). If on the other hand the behavior of students in a cluster sufficiently diverges clusters might split (this could mark the development of different learning strategies). If a distinct behavior disappears within a group of students, we assume the cluster will dissolve, meaning students will uniformly change to other clusters. In contrast, forming clusters have the potential to mark the development of distinct strategies within students.

The resulting clustering pipeline addressing these challenges is illustrated in Figure 1. The only input required are action sequences extracted from student log data. These action sequences are transformed into Markov Chains for every session and pairwise similarities between these chains are computed. Students are clustered based on these similarities while enforcing temporal consistency over consequent training sessions. Finally, we compute the optimal number of clusters for each training session.

Action Sequences. In a first step we extract action sequences $A^u_t = (a_0, a_1, \ldots, a_n)$ for every session $t$ of a user $u$. To do so, we map events in the log files of an ITS (e.g. correct/incorrect inputs or help calls) to the actions $a_i$. As the particular actions depend on the ITS, the extraction of actions has to be changed depending on the ITS.

Action Processing. While action sequences provide rich temporal information about the exact ordering of actions, we expect that they exhibit a considerable amount of noise. We therefore transform the action sequences into an aggregated representation using Markov Chain models, similar to \cite{17}. Markov Chains provide an aggregated view of the pairwise transition probabilities of actions and can be fully described by these transition probabilities $t_{i,j} := p_{a_j|a_i}$, from any state $a_i$ (in our case an action) to any other state $a_j$. Markov Chains can be extracted using maximum likelihood estimates of the transition probabilities $t_{i,j}$.

Similarity Computation. To cluster student behavior, a suitable similarity (or distance) measure between students has to be defined. In educational data mining, popular choices for measuring distances between action sequences are the longest common subsequence (LCS) and the Levenshtein distance (see e.g. \cite{4}). LCS measures the length of the largest set of characters that appear in left-to-right order within the string, not necessarily at consecutive places. The Levenshtein distance computes the number of insertions, deletions and replacements needed to transform one string into the other. Instead of computing distances directly on action sequences we can apply the computation to the aggregated values of Markov Chains. Previous work \cite{17} has been using the Euclidean distance between the transition probabilities of two Markov Chains. A potential disadvantage of the Euclidean distance is that it is not designed for the comparison of probabilities. Therefore, we propose to use metrics that are specifically designed for comparing probability distributions. Since the conditional probabilities describing a Markov Chain do not form a proper probability distribution (the entries of the transition probability matrix do not sum up to one), we compute the expected transition probabilities using the stationary distribution over the actions and compare these expected transition frequencies $t_{i,j}$ instead of the conditional probabilities $t_{i,j}$. We use two common metrics: the Jensen-Shannon Divergence and the Hellinger distance \cite{21} to compute the distances between the expected transition frequencies $t_{i,j}$ of the Markov Chains.

Clustering. Using the measures defined above we compute a pairwise similarity matrix $W^t$ for every session $t$ of the training (entries of the matrix measure how similar two students are during that particular training session). These similarity matrices can then be clustered by any standard clustering method. However, clustering students for each session individually does not make use of the temporal information available. Recently, a method for clustering evolutionary data has been proposed that accurately tracks the time-varying similarities of objects over discrete time steps \cite{27}. The method assumes that the observed similarities $W^t$ are a linear combination of the true similarity between students $Ψ^t$ and random noise $N^t$:

$$W^t = Ψ^t + N^t.$$  \hspace{1cm} (1)

Instead of performing clustering directly on $W^t$, a smoothed similarity matrix $Ψ^t$ is proposed, given as

$$Ψ^t = α^t Ψ^t−1 + (1 − α^t) W^t,$$  \hspace{1cm} (2)

where $α^t$ controls the amount of smoothing applied to the observed similarity matrix $W^t$. Under some assumptions (detailed in \cite{27}) an optimal choice for $α^t$ is

$$α^t = \frac{\sum_{i,j} \sum_{k,l} \operatorname{var}(n_{i,j})}{\sum_{i,j} \sum_{k,l} (ψ_{i,j}−1 − ψ_{i,j})^2 + \operatorname{var}(n_{i,j})}.$$  \hspace{1cm} (3)

This means that the optimal $α^t$ is based on a trade-off between the estimated noise in $W^t$ and the amount of new information that $W^t$ contains compared to previous similarity matrices. If $W^t$ exhibits a lot of noise we more heavily rely on previous observations (high $α^t$) but if we observe large
discrepancies between the previous similarity estimates and the current ones (e.g. some students show a novel behavior) we emphasize the similarities from the current session (low \(\alpha\)). Finally, we use the standard clustering algorithm K-Means to cluster the smoothed similarity matrices \(\Psi\).

Model Selection. The assumption of temporal consistency in the pairwise similarities between students does not prohibit evolution of clusters if students change their behavior over the course of the training. Such long-term drifts lead to growing and shrinking of clusters eventually, and even to dissolving and forming of clusters over time. In contrast to the original AFFECT method [27], we therefore compute the optimal number of clusters in every time step. Deciding on the number of clusters is a variant of the model selection problem, for which various different criteria exist. The Akaike information criterion (AIC) and the Bayesian information criterion (BIC) are among the most common criteria for model selection. The main difference between BIC and AIC is that the BIC penalizes the number of clusters more strongly than AIC. AICc corrects the AIC criteria for finite sample sizes. For our experiments, we used AICc as it potentially reveals more clusters, which is important for our exploratory analysis of learning data. To compute the AICc the log likelihood (LL) of the model is needed. According to [23], the LL for K-Means can be formulated as

\[
LL = \sum_i \log \left( \frac{N_c(i)}{N} \phi(x_i | \mu_{c(i)}, \sigma) \right),
\]

where \(N\) denotes the number of samples, \(c(i)\) the cluster index of sample \(x_i\) and \(N_c(i)\) the number of samples in cluster \(c(i)\). The likelihood of a sample \(x_i\) that was assigned to cluster \(c(i)\) can be computed using the probability distribution \(\phi(x_i | \mu_{c(i)}, \sigma)\), where \(\mu_{c(i)}\) denotes the centroid of the cluster and \(\sigma\) the empirical variance of the data. In our case (as suggested by [23]), the probability distributions \(\phi\) are identical spherical Gaussians. To compute the LL, we embed our data points in a Euclidean space in which the distances between the points match the similarities extracted from the action sequences. To perform this embedding, we use the method presented in [12] that transforms \(N\) objects with pairwise similarities to a \(D = N - 1\) dimensional Euclidean space. We then estimate the effective dimensionality \(\hat{D}\) of our data set as the sum of eigenvalues \(\lambda_i\) of the covariance matrix divided by the largest eigenvalue \(\lambda_1\) (see [16]):

\[
\hat{D} = \sum_i \frac{\lambda_i}{\lambda_1}.
\]

This means that the effective number of parameters \(P\) for the K-Means clustering is \(P = (\hat{D} + 1)k\), where \(k\) is equal to the number of clusters (see e.g. [23] for a derivation). Based on the LL and the estimated effective dimensionality of our data \(\hat{D}\), we calculate the AICc as

\[
AICc = -2LL + 2P + (2P(P + 1))/(n - P - 1).
\]

3. SYNTHETIC EXPERIMENTS

We analyzed the properties of our clustering algorithm using synthetic data and we compared the performance and stability of our method to previous algorithms for clustering sequential educational data. Finally, we also validated our model selection step.

Experimental setup. We simulated student actions for 80 students over 50 sessions in a simulated learning environment. Students needed to solve 20 tasks per session. Student abilities \(\theta\) and task difficulties \(d\) were simulated as part of a Rasch model [26]. Student abilities for all students were sampled from a normal distribution with mean \(\mu\) and variance \(\sigma\). Task difficulties were sampled uniformly from the range \([-3, 3]\) in agreement with the common range of task difficulties [10]. Each task \(y\) consisted of eight steps \(s_j\) that a student had to complete to finish the task (this could e.g. be letters of a word to spell, performing steps of a calculation or solving a physics problem). The probability of a student correctly solving a task was then given by the Rasch model as \(p(y) = (1 + e^{-(\theta - d)})^{-1}\). In our simulation (in accordance with many ITS) a task was correctly solved if all the sub-steps are correctly solved, which defines the probability of correctly solving a step of a task \(s_j\) to be \(p(s_j) = (p(y))^j\). Finally, a student could request help at any point in time during the training. Whether the student asked for help was sampled from a Bernoulli distribution with \(p_H\). Based on the described sampling procedure we emitted the following actions for a student: new task, help, correct, incorrect, correction, task completed. The number of sampled actions per student and session depended on the performance of the student (e.g. a student who gets every step of a task correctly completes a task after eight correct actions, whereas another student who requests help and commits an error requires more actions to complete the task).

For our experiments we simulated student groups with different behavior. For the chosen range of task difficulties, student abilities are found to be normally distributed with mean \(\mu = 0\) and variance \(\sigma = 1\) (see [10] for details). We simulated good performing students by setting \(\theta = 1\) and bad performing students by setting \(\theta = -1\). According to [2], the most frequent form of help abuse are multiple consecutive help requests. We simulated this behavior by a large probability \(p_H = 0.2\) to ask for help instead of working on the task, while normal help seeking behavior has a smaller probability for requesting help \(p_H = 0.05\). Based on these different properties we simulated four groups of 20 students as follows. Group A contains bad performing students (\(\theta = -1\) that rarely ask for help \((p_H = 0.05\)). Group B consists of bad performing students (\(\theta = -1\)) that frequently use the help system \((p_H = 0.2\)). Group C and D consist of
good performing students (θ = 1) with rare (𝑝_ไฮ = 0.05) and frequent (𝑝_ไฮ = 0.2) help requests, respectively.

Our proposed pipeline offers flexibility in the choice of the similarity measure (see Section 2). We used the Jensen Shannon divergence [21], the Hellinger distance [21] and the Euclidean distance for our experiments, and refer to these approaches as Ours_SD, Ours_HD, and Ours_EUC. To measure the influence of the different elements of the pipeline on the overall performance, we compared the proposed method to previous work on clustering of action sequences. The first approach [4] works directly on the action sequences and uses the longest common subsequences (LCS) as similarity measure. Clustering is performed using an agglomerative clustering. However, to be able to better compare clustering results we used the proposed similarity measure together with K-Means. We refer to this pipeline as LCS_KM. Similar to our method, the second approach used for comparison [17] computes the similarities between students using Markov Chains. Similarities are measured using the Euclidean distance and clustering is performed using K-Means. The pipeline for this approach is denoted by MC_EUC_KM.

**Clustering Quality & Robustness.** In a first experiment, we computed the clustering quality of the different approaches with increasing noise levels. The performance of the proposed pipeline was measured using the cluster agreement in comparison to the ground truth labels. The different noise levels were simulated by increasing the variance in student abilities σ. For the sampling of the data, Figure 2 (left) illustrates the performance of the different approaches with increasing noise. Note that the performance was computed using the correct cluster count of k = 4. Our pipeline (colored in green, red, and brown) exhibits the highest performance over all noise levels. The average agreement of our best performing pipeline (P_{Ours_HD}) is substantially higher than the average agreement of the best previous approach (P_{MC_EUC_KM}), both for a low variance (P_{Ours_HD,σ=1} = 0.82, P_{MC_EUC_KM,σ=1} = 0.53) and for noisy data (P_{Ours_HD,σ=10} = 0.45, P_{MC_EUC_KM,σ=10} = 0.34).

To investigate these differences between the approaches, we measured their performance over different numbers of clusters and different noise levels. Figure 2 (middle) illustrates the results for data with a relatively low noise level (σ = 2), while Figure 2 (right) shows the clustering quality of the different pipelines on noisy data (σ = 8). In the case of small noise in the data, all methods exhibit the best performance for the correct number of clusters (k = 4), which is a desirable property. The results demonstrate that using Markov Chains (P_{MC_EUC_KM,σ=4} = 0.44) instead of working directly on action sequences (P_{LCS_KM,σ=4} = 0.40) leads to a higher clustering quality. A further increase in performance is achieved by our proposed algorithm: The variations of our pipeline exhibit a substantially higher clustering quality (P_{Ours_EUC,σ=4} = 0.66, P_{Ours_HD,σ=4} = 0.70, P_{Ours_SD,σ=4} = 0.70) than the previous work. This substantial increase in performance (∆P_{σ=4} = 0.26 compared to MC_EUC_KM) is due to two changes in the pipeline. First, the proposed pipeline uses the AFFECT method for clustering leading to an increase in performance of ∆P_{σ=4} = 0.20. Second, while MC_EUC_KM computes the similarity measure directly on the transition probabilities, we use the expected transition probabilities as a basis for the similarity computations (see Section 2) accounting for an improvement in performance of ∆P_{σ=4} = 0.06. Within our approach, the choice of similarity measure has only a small impact on the clustering quality. Figure 2 (right) demonstrates that our proposed method is more robust to noise than previous work [17, 4]. The best variation of our pipeline (colored in green) still achieves a reasonable performance (P_{Ours_HD,σ=8} = 0.54). At these noise levels, the choice of action processing (Markov Chains vs. direct processing of action sequences) does not significantly influence performance (P_{LCS_KM,σ=4} = 0.34, P_{MC_EUC_KM,σ=4} = 0.35).

The choice of the clustering algorithm on the other hand is important. The increased performance of our method can be attributed to the use of AFFECT for clustering: AFFECT takes into account data from previous time steps to perform the clustering. Interestingly, the pipeline using the Jensen Shannon divergence (Ours_SD) seems less robust to noise than the other pipelines (Ours_HD and Ours_EUC).

**Stability.** When clustering student actions over time, temporal consistency of clusters is essential. We measured the temporal stability of our method by computing the cluster size over the 50 simulated sessions (see Figure 3). We compared the best performing pipeline from the first experiment (Ours_HD) to the previous approaches (LCS_KM, MC_EUC_KM) using again k = 4 clusters. As can be seen from Figure 3 (left), our method provides a smooth temporal clustering with stable cluster sizes over time. The clusters found by MC_EUC_KM (Figure 3 (middle)) and LCS_KM (Figure 3 (right)), on the other hand, are unstable: cluster sizes vary significantly over time. These results are as expected, as static clustering approaches identifying groups of students at each point in time are very sensitive to...
The proposed method solves this problem by applying an evolutionary clustering algorithm and therefore takes into account multiple time steps.

**Interpretability.** Since we are clustering student behavior over multiple sessions, we expect the number of clusters and the cluster sizes to change over time. We expect clusters to merge, split, dissolve and form (see Section 2 for details). We evaluated the *Ours_HD* pipeline on four scenarios using synthetic data. Note that these scenarios are artificial and are used only to demonstrate that the pipeline can capture the described events; we will show real-world examples of these events in Section 4. In the first scenario (Figure 4 (top left)), group A consisting of bad performing students with rare help calls (colored in dark green) merges into group B (colored in dark blue), i.e. the students of group A also start abusing the help. In our simulation, we start the cluster merge after \( t = 20 \) sessions and let group A completely vanish after \( t = 50 \) sessions, a behavior that is nicely captured by our method. The second scenario (Figure 4 (top right)) starts with only three groups (B, C, and D), assuming that all bad performing students frequently use the help. Over time, the bad performing students split into a group abusing the help (group B, colored in dark blue) and a cluster consisting of students with rare help calls (group A, colored in dark green), i.e. in the simulation some of the bad performing students stop abusing the help over time. In the third scenario (Figure 4 (bottom left)) a dissolving cluster is simulated: Over time, group B (colored in dark blue) completely dissolves and the students are distributed to the other three clusters. The fourth scenario (Figure 4 (bottom right)), finally, simulates a forming cluster event. The simulation starts with only three clusters (groups A, C, and D). With an increasing number of sessions, a fourth cluster forms (group B, colored in dark blue) and students from the other three clusters slowly switch to the new cluster until all the groups have equal size (after \( t = 50 \) sessions). This event is again correctly captured by our method. The presented experiments demonstrate that the proposed pipeline is able to reliably identify changing cluster numbers and sizes. The results also demonstrate the validity of the model selection step of the pipeline: The AICc correctly identifies the number of clusters for all scenarios.

**4. EXPLORATORY DATA ANALYSIS**

We applied our method to clustering of student interactions from two different ITS, focusing on the identification and interpretation of cluster events.

**Experimental Setup.** The first data set contains log data from 106 students and was collected using *Orthograph*, a computer-based training program for elementary school children with dyslexia [9]. *Orthograph* consists of one main learning game, where children have to type a dictated word. The second data set contains data from 134 students and was collected from *Calcularis*, an ITS for elementary school children with difficulties in learning mathematics [13]. *Calcularis* consists of different games for training number representations and calculation. For all students, we extracted the first 15 training sessions with a minimal duration of \( t = 5 \) minutes from each student.

All results have been computed using our pipeline *Ours_HD* (see Section 2), applying the Hellinger Distance to measure similarities between Markov Chains of different students.

**Navigation Behavior.** In a first experiment, we extracted actions describing the Navigation Behavior of children in *Orthograph*. Navigation Behavior captures all events that cause the displayed content to change. During game play, children collect points for correct responses as well as for time spent in the training in general. These points can be used to buy different visual perks for the game in the shop. Children can also analyze their performance (e.g. progress in the current module) in the progress view. The resulting Markov chain (see Figure 5) consists of three possible states: *Game*, *Shop*, and *Performance*.

Figure 6 shows the relative cluster sizes for the Navigation Behavior Markov Chain over the first 15 sessions of the training. The different colors denote different clusters. At the beginning of the training (\( t = 0 \)), our pipeline detects seven different clusters, however, three of these clusters (col-
ored in pink, brown, and orange) die within the first three training sessions. Children in these clusters spent more than 50% of their time browsing the shop and checking their performance (orange: 46% Game, 31% Shop, 23% Performance; brown: 43% Game, 22% Shop, 35% Performance; pink: 40% Game, 32% Shop, 28% Performance) at the beginning of the training. We therefore hypothesize that children in these clusters tried out and played with the different views before getting used to the navigation possibilities of the system.

After $t = 5$ time steps, a further cluster (colored in green) dissolves before the clustering stabilizes to three main groups (colored in blue, red, and purple). Figure 7 (top) shows the transition probabilities of the Markov Chains for the different clusters before the clusters dissolve (after $t = 3$ sessions). Children in the blue cluster are very focused on training, they spend 82% of their time in the Game. Once in the Shop or Performance state (18% of their time) they tend to select the following view with equal probabilities. Children in the red cluster like to browse the shop, a behavior that is visible from the high transition probabilities to the Shop state (Game→Shop: 0.41; Performance→Shop: 0.39), resulting in 34% of the training time spent browsing the shop. The purple cluster consists of children, who like to navigate to the shop and performance overview between solving the different tasks (Game→Shop: 0.41, Game→Performance: 0.44). However, these tend to be short visits as they will return to playing the game right after with high probability (Performance→Game: 0.58, Shop→Game: 0.77). Finally, children in the green cluster tend to select the next view randomly when playing the game. Once in the Performance state, they have a probability of 0.30 to browse the shop right after. The analysis of this time step illustrates that the different clusters differentiate well between focused children not making use of the navigation possibilities (blue cluster), children who frequently (but reasonably) use the different views (purple and green cluster), and distracted children who spend long amounts of time off-task (red cluster).

After $t = 6$ training sessions, the green cluster dissolves and students from this cluster change to the red and blue clusters. The transition probabilities of the Markov Chains for these stable main clusters are illustrated in Figure 7 (bottom). The children in the blue cluster are still focused on training, spending 76% of their time solving tasks. However, they also check their training progress from time to time (14% of the time spent in the Performance state). After checking training progress, they tend to also browse the shop (Performance→Shop: 0.27). The children in the purple cluster have stopped navigating to the performance overview between different tasks (Game→Performance: 0.17) and instead visit the shop more frequently (Game→Performance: 0.58) and longer (35% of time spent in the Shop state). The red cluster still consists of children who like browsing the shop, a behavior that is visible from the high transition probabilities to the Shop state (Game→Shop: 0.33; Performance→Shop: 0.31). However, they also tend to spend time checking their progress, resulting in 47% of the training time spent off-task. Students from the green cluster therefore changed their behavior from frequent, but short off-task navigation to a more focused training style (change to blue cluster) or to being completely distracted and spend long amount of times off-task (change to the red cluster).

**Input & Help Seeking Behavior.** Our method can be used as a black box for any ITS and therefore also allows for comparison of behavior patterns across different ITS. The only user input needed is the definition of possible actions. To illustrate this possibility, we extracted two different sets of actions Input Behavior and Help Seeking Behavior from data collected with Orthograph and with Calculars.

**Input Behavior** captures all possible inputs. Implicitly these actions capture the performance of students, as e.g. a bad performing student is likely to commit more mistakes. In Orthograph, children train spelling by writing words that are played back by the system. Therefore, the Input Behavior Markov Chain for Orthograph (see Figure 8) consists of four states: Children can type a letter (Input), correct themselves by deleting a letter (Backspace), provide invalid input such as typing a number (Invalid Input), or submit their solution (Enter). For Calculars, we investigated calculation games. In these games, children need to solve different mental addition and subtraction tasks. We again define four states for the Input Behavior Markov Chain (see Figure 8): children can type a digit (Input), correct themselves by deleting a digit (Correction), provide invalid input such as random mouse clicks (Invalid Input), or set their answer (Enter).

Figure 9 shows the relative cluster sizes for the Input Behavior action set from Orthograph over 15 training sessions. Our method identifies three stable clusters. Investigating the stationary distributions of the Markov Chains reveals that students in the orange cluster show the highest probabilities for committing invalid inputs over all sessions ($t = 3$: 0.15; $t = 7$: 0.23; $t = 13$: 0.16). The green cluster consists of focused students who consistently produce a low percentage of invalid inputs ($t = 3$: 0.06; $t = 7$: 0.04; $t = 13$: 0.05). Students in the blue cluster also tend to show low probabilities for invalid inputs across the different sessions ($t = 3$: 0.11; $t = 7$: 0.09; $t = 13$: 0.08). The orange cluster is an example of a forming cluster growing in size over the course of the training. We hypothesize that this event marks the increasing difficulty of the tasks and is caused by a downwards drift.
of students from the clusters with good performing students to the clusters with students showing worse performance. Further analysis of cluster transfers reveals that students indeed are never switching directly from the green (best performance) to the orange cluster (worst performance).

For Calcularis, the Input Behavior clusters are relatively stable over the course of the training (see Figure 9). There is one distinct dissolve event in the first four sessions: the orange cluster is dissolving into the blue and green clusters. Investigating the stationary distributions of the Markov chains of the three clusters reveals that all clusters have a relatively low probability for invalid inputs (t=2: 0.17 (blue), 0.12 (orange), 0.08 (green)). However, students belonging to the blue cluster tend to perform multiple consecutive corrective actions in a row (Correction→Correction: 0.25 (blue), 0.13 (orange), 0.13 (green)). Students in the orange cluster are most likely to enter a valid input after a correction (Correction→Input: 0.68 (orange), 0.57 (blue), 0.65 (green)).

In Orthograph, differences in Input Behavior are mainly expressed by the percentage of invalid inputs provided. We observe a more distinct picture for Calcularis. While the invalid inputs are still an important indicator, children also exhibit different corrective behaviors.

Help Seeking Behavior captures the use of hints available in the training environment. In Orthograph, children can re-play the given word (Hear Word), play the melody of the word (Play Music) and show the correct spelling of the word (Show Word). The according Markov Chain is displayed in Figure 8. The states New Task and Input denote the play-back of a new word and a user input (keyboard), respectively. The development of the relative cluster sizes for these action sequences (see Figure 9) reveals a surprisingly large variance in student behavior (the clustering algorithm finds nine different clusters in the first two training sessions). However, the diversity in student behavior disappears through a large cluster merge after t=3 sessions. Investigating the transition probabilities between the different actions, we observe that while students are experimenting with the three different help systems at the beginning of the training, the final cluster of students gave up on using the help functions. This drop in the frequency and diversity of help usage indicates that the help functionality provided in Orthograph is not useful for most of the students.

Calcularis provides a limited help functionality. Children can require explanations for games (Help). Furthermore, they can directly require the solution of a task (Empty), if the task seems too difficult. Further states of the Markov Chain (displayed in Figure 8) are the setting of a complete answer (Regular) and the abortion of a task (Incomplete). We again observe a large cluster merge at the beginning of the training leading into two stable clusters. Investigating the stationary distributions of the Markov Chains of the two clusters reveals that students in the orange cluster are more likely to perform a help request compared to the blue cluster (t=6: 0.03 (blue), 0.13 (orange)).

The Help Seeking Behavior of the children is more difficult to compare across different ITS, because the available hints are very different. However, our experiment shows that both learning environments do not provide ideal help options.
Figure 9: Relative cluster sizes over the first 15 sessions based on the clustering of Input Behavior (left) and Help Seeking Behavior (right) for students training with Orthograph and Calcularis.

5. CONCLUSIONS
We presented a complete pipeline for the evolutionary clustering of student behavior. This pipeline can be used as a black box for any ITS, requiring only the extraction of action sequences as input. We demonstrated that enforcing temporal coherency between consecutive clusterings is beneficial for the detection of student behavior as well as the stable detection of cluster events. Our method outperforms previous work on synthetic data regarding clustering quality and stability. We applied our pipeline to different types of action sequences collected from two different ITS. The exploratory analysis demonstrates that our method is able to reveal interesting properties about the behavior of students and potential deficiencies of the learning environments.

Acknowledgments. This work was supported by ETH Research Grant ETH-23 13-2.

6. REFERENCES
Web as a textbook: Curating Targeted Learning Paths through the Heterogeneous Learning Resources on the Web.

Igor Labutov
Cornell University
iil4@cornell.edu

Hod Lipson
Columbia University
hod.lipson@columbia.edu

ABSTRACT
A growing subset of the web today is aimed at teaching and explaining technical concepts with varying degrees of detail and to a broad range of target audiences. Content such as tutorials, blog articles and lecture notes is becoming more prevalent in many technical disciplines and provides up-to-date technical coverage with widely different levels of prerequisite assumptions on the part of the reader. We propose a task of organizing heterogeneous educational resources on the web into a structure akin to a textbook or a course, allowing the learner to navigate a sequence of web-pages that take them from point A (their prior knowledge) to point B (material they want to learn). We approach this task by 1) performing a shallow term-level classification of what concepts are explained and assumed in any given text, and 2) using this representation to connect web resources that explain concepts to those web resources where the same concepts are assumed. The main contributions of this paper are 1) a supervised classification approach to identifying explained and assumed terms in a document and 2) an algorithm for finding optimal paths through the web resources given the constraints of the user’s goal and prior knowledge.

Keywords
web resources; optimizing learning

1. INTRODUCTION
No scholar is born at the frontier of knowledge — early learning and lifelong learning both play a defining role in shaping the research vector of an academic [7]. More alarming, recent research [6] demonstrates that the pre-career idle time of an up-and-coming researcher has been on the steady rise during the last century, attributing to the “burden of knowledge” phenomenon — the inflation of the body of prerequisite prior knowledge to be mastered before being able to contribute to the field with original research. The hypothesis of [9, 10] is that facilitating effective early and lifelong learning practices is a viable way for easing the “burden of knowledge”.

While physical textbooks and classrooms traditionally assumed the role of knowledge curators, they also present a bottleneck in today’s rapidly growing web of up-to-date technical and academic content — peer-reviewed articles, lecture notes, tutorials, slides etc — from academics and “citizen scientists” alike. An automatic approach for “weaving” natural curricular progressions through the web of such heterogeneous academic/educational content, we believe, will catalyze early and lifelong learning by creating more efficient and goal-oriented curricula targeted to the level of the audience.

The web is the only collection of resources today where attempting this task becomes meaningful and promising. The reason for this is that the web contains an extensive amount of diversity in its content, i.e. content that explains the same concepts but in many different ways. Naturally this diversity reflects the diversity of the people who create this content, their backgrounds, styles of learning and ways of thinking about complex concepts, which would naturally match learners with similar characteristics. We believe that this diversity can be leveraged to create learning pathways that are not bound to the traditional curricula that are often constrained for no better than a historical reason. We propose instead to optimize a curriculum directly for what you want to know given what you already know.

We propose to tackle the problem of curriculum mining on the web, which broadly, involves linking technical resources on the web to other resources that explain a subset of concepts that are assumed in the original document. We propose to decompose the task into 1) understanding what is explained and assumed in a document on the part of the the reader and 2) use this document-level representation to sequence documents that guide the learner from their current state of knowledge towards their goal, for example, understanding a specific research paper or a set of lecture notes.

We propose a term-centric approach for inducing curricular relations between any pair of documents. Naturally, understanding a technical concept is more than being familiar with its surface term, and in this view an approach that operates at the level of individual terms may appear to be naïve. After all, to explain a new concept is to put together existing concepts in a novel way [13], and in the process introduce convenient nomenclature. However, we hypothesize, that by the virtue of seeking the shortest sequence of documents that “cover” (explain) multiple terms at once, the resulting bottle-
neck will implicitly “prefer” to link to prerequisite documents that introduce and explain whole concepts, i.e. groups of terms, as opposed to introducing terms one document at a time (an extreme example would be presenting a sequence of pages from a dictionary, each document defining a term independence; this is clearly undesirable). It will be our running assumption, that there exists a correlation between the knowledge of the terms and the understanding of the overarching concept.

Thus, to a first-order approximation, we model technical documents as “bags of terms”, and in the interest of tractability set forth the following set of modeling assumptions:

- **Assumption 1** A document is a bag-of-technical-terms (multiset) that is further partitioned into two multisets: \( E \) (Explained), \( A \) (Assumed) — corresponding to the role (aspect) of the term within the document:
  - **Explained**: The terms appear in the context that further the understanding of the concept corresponding to the term.
  - **Assumed**: The concept corresponding to the term is assumed to be familiar, and is required for understanding the context in which it appears.

- **Assumption 2** The degree of reliance on the knowledge of a particular term in the document is proportional to the frequency of the term in the Assumed multiset, i.e. which concepts are fundamental to the understanding of the document, and which are auxiliary is reflected in the number of occurrences of the corresponding terms.

As an illustration, consider the following excerpt from Christopher Bishop’s classic textbook *Machine Learning and Pattern Recognition* from the chapter that introduces the concept of **Expectation Maximization**:

**Expectation Maximization**

An elegant and powerful method for finding maximum likelyhood solutions for models with latent variables is called the expectation maximization algorithm, or EM algorithm.

In the excerpt above, we solid-underline the terms that appear in the Explained aspect and dash-underline terms that appear in the Assumed aspect. Understanding the concept of **Maximum likelihood** is a prerequisite for understanding **Expectation Maximization**. It is no surprise that most resources that introduce the concept of **Expectation Maximization** implicitly assume that the reader is familiar with **Maximum Likelihood**. Academic and educational literature is fraught with such implicit assumptions that may be challenging to unravel for a learner especially new to the area. Note that on the surface it may seem that detecting instances of explained terms in the text is an equivalent task to finding instances of term definitions – a well studied task – but it is not so. Especially in technical disciplines, explaining a concept requires much more than giving a definition. A document defining a term, may or may not actually explain the concept behind it. For example, a document may define a term to refresh the reader’s memory but otherwise assume the reader’s familiarity with it. On the other hand, a document may explain a term without ever giving a one-sentence definition.

Finally, the proposed dichotomy may appear as a gross oversimplification, ignoring the entire continuum of pragmatics between the two extremes. We argue that while binary term-level classification alone may not capture the fine-grained aspect of any one term, combining it with the context of the entire document, will enable us to unravel the prerequisite relationships between documents.

2. RELATED WORK

Evidence of information overload in traditional textbooks

Formal study of textbook organization conducted by [1] on a corpus of textbooks from India quantitatively addresses the issue known as the “mentioning problem” [12], where “concepts are encountered before they have been adequately explained and forces students to randomly ‘knock around’ the textbook”. The work of [1] suggests that many traditional textbooks suffer from the resulting phenomenon of “information burden” and provide diagnostic metrics for evaluating it. A user study conducted by [2], though limited to electronic textbooks, demonstrated the utility of a navigational aid that links concepts and terms within a textbook and allows the user to navigate according to own preferences. This suggests the potential utility of tools that expand such “navigational ability” outside textbooks.

Attempts at manual curriculum curation

There have been at least two efforts that we are aware of, that attempts to manually create “paths” between a selected set of resources on the web — two educational start-ups, Metacademy [5], and Knewton [4]. While motivated by the same goal, we believe that manual web-scale curriculum curation is akin to the manually-curated directory of the web (not too different from the original Yahoo directory from the 1990s), i.e. offering poor scaling capability in the dynamic, growing landscape of educational content on the web.

Attempts at automatic curriculum curation

Most relevant to our task is the work of [11] that attempt to infer prerequisite relationships between a pair of Wikipedia articles. They frame the problem of prerequisite prediction as “link-prediction” between a pair of pages using primarily graph-derived (e.g. hyperlink structure) and some content-derived features (e.g. article titles). In contrast to their approach, we do not assume any existing structure connecting the web resources (e.g. within Wikipedia), as the majority of the educational content on the web is unstructured. Our approach also naturally facilitates a scalable assimilation of new content, as we require only a document-scoped term-level classification, without needing to explicitly construct or update a prerequisite graph. Furthermore, we develop an approach for optimizing curricular paths using the proposed representation. More recent work of [8] develop a method that does not rely on a manual annotation of the prerequisite relations as in [11], and instead uses the statistics of concept reference in a pair of pages to determine the prerequisite relation between them. Similar to [11], their focus is on the pairwise link prediction, in contrast to our goal of globally optimizing a learning curriculum.
3. MODEL

3.1 Modeling explanations

We model the problem of identifying the explained and assumed terms in a document as a term-level binary classification task, i.e. each term in the document is classified into one of the two categories. Although simple from an implementation perspective, this task is made difficult by the lack of annotated data in this domain. In this work, we rely on (i) manual annotation of the term aspects performed by us for one of the textbooks (Rice University’s statistics text) and (ii) explicit annotations from the index of Bishop’s Pattern Recognition and Machine Learning textbook that were made by the author of the text (the annotation is in the form of a location in the text where a particular concept is explained).

The Rice University’s Online Statistics Education: An Interactive Multimedia Course of Study textbook, from hereon referred to as STATSBOOK consists of a total of 112 units, with a median of 12.5 unique technical terms per unit, for a total of 339 different technical terms in the book. We scrape the text content of the book from the web, replace all mathematical formulae and symbols with special tokens, and manually annotate each technical term mention with its representative form from the index, i.e. normally distributed with normal distribution. Manual term annotation obviates the need for introducing a word-sense disambiguation component and additional errors. We process the PRML dataset in an identical manner.

Each technical term in every unit of the book was annotated with the binary {explain, assume} aspect, following the definitions outlined on the previous page. While for most terms, the application of these definitions is fairly unambiguous, for a significant number of term mentions, the aspects are not mutually exclusive, i.e. the term may be construed to belong to both aspects simultaneously. Often, in using (assuming) a term to explain a related concept, something about the assumed term is also explained as a side effect. The degree to which the explanation is distributed between the terms is difficult to judge objectively, and may vary between distinct mentions of the terms in different parts of the same document. We adopt a simple strategy for “breaking ties” in such cases: if we a judge a term as having been intended to be explained in the given context by the author, we mark it with the explain aspect, otherwise, the term is assumed to be assumed. In total across the entire STATSBOOK corpus, 1878 terms were annotated for their aspect (note that the same term appears in multiple documents with potentially different aspects), with a class ratio of 537 terms belonging to the explain and 1341 terms belonging to the assume aspect.

The PRML dataset contains a total of 3883 annotated terms, with 222 terms belonging to the explain and 3661 terms belonging to the assume aspect. The aspect of the term was determined from the index of the book, which explicitly specifies the pages where a term is explained.

A logistic regression model (LIBLINEAR [3] with default regularization parameter) was trained to predict a binary aspect of the terms and evaluated with 10-fold stratified cross-validation. A set of lexical and dependency features describing the context of each term (within a 1 sentence window), positional features describing the location of the term’s mention within the document and sentences in which the term appeared, and the frequency rank of the term within the document were employed. We compare the performance of a classifier that uses all of these features with the one that uses only the rank. A classifier that is given rank as the only feature, will essentially learn a rank “threshold” that will decide the aspect of the term within the document, i.e. predict all terms above a certain rank as explained.

Figure 1(a) summarizes the performance of aspect prediction with the classifier trained using both linguistic and rank features (Rank+Text, AUC=0.76) versus a classifier trained using only the rank (Rank only, AUC=0.66) for the STATSBOOK corpus. As expected, rank is predictive of the aspect, but contextual linguistic cues provide a significant boost.

Keeping our end goal in mind, under Assumption 2 stated in the introduction, we hypothesize that the frequency rank of the term in a document correlates with the degree to which a term is either assumed or explained in that document. In the downstream task of linking documents to their prerequisites, getting the aspects of the more frequent terms correct is arguably more important than of the terms that only appear once or twice. We evaluate the performance of our aspect classifier as a function of the term’s rank. Figure 1(b) illustrates predictive performance (AUC) on a subset of the data stratified by the term’s frequency rank. We observe a favorable trend in increased predictive performance for higher ranked terms. An obvious explanation is that more frequent terms accumulate a larger set of features describing them (since each mention of the term contributes its context features), effectively decreasing variance in the predictions.

3.2 Optimal learning paths

Consider now that we have a large collection of documents (e.g. tutorials, papers, textbook chapters). Each such document explains some concepts but also assumes the reader’s knowledge of other concepts (e.g. a tutorial may explain the concept of normal distribution, but may assume the knowledge of probability and distribution). We will now consider that we can reliably classify each term in each document into either the Explained or Assumed category. Consider that we also have a user who is interested in understanding a specific (target) document (or a set of target documents). The goal is to give a user a self-contained sequence of documents of
We say that the document $d$ is covered when every one of its assumed terms is explained by at least one document in the prerequisite set. The goal is to find a smallest set of documents such that it covers a user’s document of interest to the user. Figure 2 illustrates a feasible solution, where each document is covered.

Formally each document $d_i$ in our collection is a set of two sets of terms: the explained terms $E_i = E(d_i)$ and the assumed terms $A_i = A(d_i)$. A term in any document is either explained or assumed, but not both, i.e. $A_i \cap E_i = \emptyset$. We say that the document $d_i$ is covered by a prerequisite set of documents $P_i$ when:

$$A_i \subseteq \bigcup_{d_j \in P_i} E(d_j)$$

In other words the document is covered when every one of its assumed terms is explained by at least one document in the prerequisite set. For any prerequisite set that covers this document, the documents in the prerequisite set need to be covered as well, recursively until all documents have been covered. We assume the existence of documents with no prerequisites (leaves), i.e. those documents for which $A_i = \emptyset$.

The goal is to find a smallest self-contained set of documents $P_i$; i.e. a set of documents such that all the documents in $P$ are covered and $d_0 \in P$, where $d_0 = \{A_0, E_0\}$ is the target document of interest to the user. Figure 2 illustrates a feasible solution to an example problem. Without additional restrictions, solutions to this problem can contain cyclical dependencies. Such cycles don’t make sense in our setting. Thus an important restriction is that the the set of documents $P$ can be ordered such that every document in the sequence is covered by the preceding documents in the sequence. Let $p$ be a sequence of documents of length $K$, where $p_k$ is the $k^{th}$ document in the sequence, then we seek:

$$\text{minimize} \ |p|$$

s.t. $\forall k : A(p_k) \subseteq \bigcup_{k' = 0}^{k-1} E(p_{k'})$

$$d_0 \in p$$

(1)

**ILP formulation**

We formulate an Integer Linear Program (ILP) that finds a minimum length self-contained sequence $p$ of at most $K$ documents such that it covers a user’s document of interest $d_0$. Consider that we have a total of $D$ documents. We define the following variables:

$$x_{k}^{i} \in \{0, 1\} \quad \text{document } d_i \text{ is in } k^{th} \text{ position in the sequence}$$

We define the following constants:

$$e_{ij} \in \{0, 1\} \quad \text{Term } j \text{ is explained in document } i$$

$$a_{ij} \in \{0, 1\} \quad \text{Term } j \text{ is assumed in document } i$$

Each assumed term in a document in position $k$ must be explained by at least one document up to (but not including) the document in position $k$. This can be expressed via the following constraint:

$$\sum_{k' = 0}^{k-1} e_{ij} x_{k'}^{i} \geq \sum_{k' = 0}^{D} a_{ij} x_{k'}^{k} \quad \forall j \forall k$$

Each position in the sequence contains at most 1 document:

$$\sum_{i=0}^{D} x_{k}^{i} \leq 1 \quad \forall k$$

User’s preference of covering a document of interest $d_0$ is an additional constraint:

$$\sum_{k=0}^{K} x_{k}^{0} = 1 \quad \forall k$$

Finally, the objective is to minimize the number of documents in the sequence:

$$\text{minimize} \sum_{k=0}^{K} \sum_{i=0}^{D} x_{k}^{i}$$

The above formulation also allows us to directly incorporate the user’s prior knowledge into this optimization problem. If we represent a user as a set of explained terms, i.e. terms that the user is assumed to have mastered, then the constraints corresponding to these terms may simply be dropped from the formulation.

In the most general case, this formulation has $D^2$ variables and $O(D^2 \times V)$ constraints, where $V$ is the number of terms in the vocabulary. In practice, however, we will often limit the maximum allowable sequence length to a fairly small constant (e.g. 10, as done in our experiments), reducing the order of the problem to $O(D)$ variables and $O(D \times V)$ constraints.

While in extremely large settings (hundreds of thousands of documents), even with a small $K$, solving this ILP directly is infeasible, in practice, we find that that we can obtain exact solutions using LP relaxation and a vanilla Branch and Bound (using GLPK) within several seconds, even with a many as 1,000 documents and hundreds of terms. Developing an approximation algorithm based on rounding the LP solution is our ongoing work.

https://www.gnu.org/software/glpk/
4. EVALUATION

4.1 Prerequisites

In order to evaluate the Explain/Assume classifier in an end-to-end setting, we employ the output of this classifier in the task of predicting prerequisites in a dataset where the prerequisites have been explicitly annotated. One such resource is Rice University’s Online Statistics Textbook, which in addition to the text content, provides an explicit dependency graph annotating prerequisite relations between pairs of units (units are at the level of chapter sections). We propose a metric for scoring a pair of units according to their prerequisite relationship based only on the terminology of both units and the output of the Explain/Assume classifier.

The proposed “prerequisite score” is defined as follows:

\[
P(d_a \rightarrow d_b) = \frac{\sum_{t_i \in d_b} n^a_i \mathbb{I}[t_i \text{ explained in } d_a]}{\sum_{t_j \in d_a} n^a_j \mathbb{I}[t_j \text{ explained in } d_a]}
\]

where \(n^a_i\) is the number of occurrences of term \(i\) in document \(d_a\). Since the above score is guaranteed to be in the \([0, 1]\) range, we can interpret it as a probability \(P(d_a \rightarrow d_b)\), a probability that document \(a\) is a prerequisite of document \(b\). There is an intuitive interpretation to the above score: a document can be considered a strong prerequisite of a target document when it explains all of the assumed terms in the target document and nothing more. We can convince ourselves that in this case the score as defined above will be equal to 1. A document that explains too many unrelated concepts will suffer a penalty with respect to its prerequisite score to another document. Furthermore, we consider the relative frequency of the explained term in the prerequisite document as an additional signal of that term’s importance. We find that this additional information increases the performance of prerequisite classification (discussed at the end of this section).

Because the output of the Explain/Assume classifier is a probability, rather than a class, we can relax the above score to directly incorporate the uncertainty in the classification:

\[
P(d_a \rightarrow d_b) = \sum_{t_i \in d_b} n^a_i P(t_i \text{ explained in } d_a)
\]

Note that in addition to relaxing the requirement of an explicit Explain or Assume label, we also drop the requirement that only the assumed terms need to be explained to count towards the prerequisite score. This distinction is optional, but it encodes an important assumption on the kinds of “prerequisites” that this score will discover. This also brings up the importance of being precise about the definition of a prerequisite. A document \(a\) is a strict prerequisite of document \(b\), if document \(a\) explains a subset of the assumptions in document \(b\). However, we can relax this definition by not requiring that the terms explained in the prerequisite (\(a\)) are strictly assumed in the target (\(b\)). In other words, a document that explains a subset of the terms also explained in the target and nothing else, will have a score of 1 according to the above equation. In practice this corresponds to documents that explain the same concepts but in a simpler way (since they explain only a subset of the explained concepts in the target), and this is often a desired behavior in a learning sequence. For example, before reading a more advanced article on Support Vector Machines, the learner might want to read a more basic introduction to Support Vector Machines, although from the perspective of term classifications, both documents explain the same concept.

4.1.1 Reconstructing prerequisites

Rice University’s Online Statistics Textbook provides a valuable resource for evaluating the effectiveness of the Explain/Assume classification at the task of predicting prerequisite relations between documents. The textbook consists of 112 units at the granularity of chapter sections, annotated as a directed graph, i.e. specifying a directed edge between a pair of units if one unit is considered a prerequisite of another unit. We process the raw HTML files of the textbook by removing markup, segmenting sentences and extracting terminology (obtained from the index) features as described in Section 3.1. We pose the problem of prerequisite relation prediction as a standard binary classification task, i.e. predicting for each pair of units in the book whether one unit is a prerequisite of another, where we consider a pair of units to be in a gold-standard prerequisite relation if there is a directed path between them in the graph. AUC is a convenient metric for evaluating performance in this prediction task, as the output of our scoring metric (Equation 2) is already scaled between 0 and 1. Note that the model trained only on the PRML corpus was used for term-aspect classification in this task. Figure 3 illustrates the results for three different models, as a function of the prerequisite depth, i.e. stratifying the classification results for a pair of units by the maximum distance between them in the graph. The three models evaluated are as follows:

- **Model** Prerequisite score is computed with Equation 2.
- **Baseline 1** Prerequisite score is computed with Equation 2, but with all \(n^a_i, n^b_i\) and \(P(t, \text{ explained in } \cdot)\) set to 1. This baseline is equivalent to a ratio between the number of overlapping terms between a pair of documents and the number of terms in the prerequisite, i.e. \(\frac{|d_a \cap d_b|}{|d_a|}\).
Baseline 2 Prerequisite score is computed with Equation 2, but with $P(t \cdot \text{explained in } -)$ set to 1.

Each baseline illustrates the effect of not including a component of the scoring function in Equation 2. Our first conclusion from the results in Figure 3 is that the output of the Explain/Assume classifier provides an important signal in predicting the prerequisite relationship between documents. Furthermore, the relative frequency of the explained terms in the prerequisite document provides an additional gain in performance. This can be explained by Figure 1(b): the performance of the Explain/Assume classifier is greater in the higher term-frequency regime; discounting low-frequency terms (that are also likely less important to the content) reduces the classification noise and boosts the performance at the prerequisite prediction task. An additional observation is that the performance of the pairwise prerequisite classification improves for pairs of units that are closer in the graph, i.e. with less units in between. This is easily explained: units that are farther apart typically share less terminology, making the estimates based on terminology overlap noisier.

It is also interesting to note that the simplest baseline that considers only the ratio of overlapping terms between a pair of documents to the total number of terms in the prerequisite document does surprisingly well, especially well for pairs of documents closer together. This can be explained as follows: in a sequence of units like those in a textbook, units that are prerequisites tend to be less advanced, i.e. have less terminology, since less of it was introduced up to that point. Thus, units that are prerequisites, at least in a textbook, would be fairly predictable from the relative frequency of overlapped terms alone.

4.2 Scaling to the web

We collect and release two web corpora of educational content in the areas of Machine Learning and Statistics. Both corpora were collected using Bing Search API, by querying for short permutations of terms collected from the index of the Pattern Recognition and Machine Learning and Rice University’s Online Statistics Textbook. The two corpora contain 42,000 and 1,000 documents respectively – a mixture of HTML and PDF files, pre-processed and converted to plain text. The difference in size of the two corpora is due to a smaller set of keywords used in the query set, and used primarily to rapidly validate the proposed model for path optimization. Consequently, because of a smaller term vocabulary, the smaller corpus is significantly less noisy (less irrelevant documents). The union of the terminology from the index of both textbooks was used as the vocabulary in processing each document. Additionally, terminology variations and abbreviations were consolidated using the link data from Wikipedia, e.g. terms EM, E-M, Expectation-Maximization, are all mapped to the same concept of EM in the terminology extraction stage.

Following the extraction of terminology from each webpage, each term is classified using the Explain/Assume classifier trained on the Pattern Recognition and Machine Learning textbook. We train this classifier in a fully supervised setting using all of the annotated data. In the next several sections, we present the analysis of the two web corpora and demonstrate the effectiveness of the proposed approach to connecting educational resources on the web.

4.3 Diversity of assumptions

The web is a unique setting, that unlike a traditional textbook or a course, offers a multitude of diverse explanations of the same concept. This diversity potentially enables the level of personalization that is not possible in traditional resources. We can analyze the diversity in the educational content on the web by looking at a slice of the web resources that share the same topic, but differ in their underlying assumptions and explanations. Figure 4 illustrates two articles that are both on the topic of Expectation Maximization. However, the two articles differ significantly in their assumptions on the background of the reader. Article 1 (left in Figure 4) is a very basic introduction to the topic and does not assume the knowledge of even the concept of maximum likelihood, which under most traditional curricula is assumed to be the prerequisite. Article 2 (right in Figure 4), however, assumes the knowledge of many more concepts such as posterior probability, likelihood function and maximum likelihood. This difference in the distribution of the underlying assumptions is explained by the fact the Article 1 s a very basic introduction to the topic, intended for an audience not in the area of statistics or machine learning. Article 2, however, is a significantly more thorough and a more technical introduction to the concept of the Expectation Maximization algorithm and thus assumes significantly more prerequisite background in the areas of statistics and machine learning. It's important to note that this distinction between the two documents cannot be easily made from their titles, or other surface cues: both documents are approximately the same length and their titles do not give away the level of technical detail. Their text content, however, provides the necessary cues to this information.

4.4 Fundamental prerequisites

Figure 5 illustrates the result of optimizing a learning path over the web corpus of 1,000 documents for the target webpage on the topic of “Maximum Likelihood Estimation”. Sequences were optimized using the ILP formulation described in Section 3.2 using the GLPK Branch and Bound solver. Red rectangles correspond to terms for which the predicted label is assumed in the given document, and blue otherwise. In addition to the term-coverage diagram, we also illustrate the prerequisite dependencies extracted from the term coverage data: a directed edge is drawn to a document from the closest prerequisite in the sequence that covers at least one assumed term in the document. In the example in Figure 5, the target web-page is a fairly technical article on Maximum Likelihood Estimation that assumes the reader’s understanding of the concepts such as the likelihood function which is pivotal for understanding the concept of maximum likelihood. As a consequence, the web-page that is placed immediately before in the optimal sequence are slides which consist of a more basic introduction to the maximum likelihood. Furthermore, the original target article assumes the reader’s familiarity with Generalized Linear Models (which is in fact the previous section of the lecture notes of that series, indicating it as a prerequisite). The resulting sequence also contains an additional prerequisite on this topic. Finally, an interesting observation is that while the target article is fairly advanced in its assumptions about the reader’s knowledge of
probability, it actually goes into surprising depth in explaining the concept of a derivative and maximizing a function using derivatives from scratch, which is another important prerequisite to the concept of maximum likelihood. This is highly unconventional in traditional textbook and course curricula. This again underlines the advantage of working with the assumptions at document-level, allowing to leverage the diversity in explanations to find “shortcuts” through the learning paths.

Figure 6 provides additional insightful examples of the generated sequences extracted from the term-coverage data of each sequence. Figure 6(d) is another example where the target document is a fairly advanced introduction to the topic (Expectation Maximization), which is preceded by a more gentle introduction to the same topic, as well as an additional prerequisite (Maximum Likelihood) which is a common prerequisite for this topic. Note, however, that while Maximum Likelihood is traditionally considered as a prerequisite for learning about Expectation Maximization, it is not the case for the more basic introduction to this topic (What is the Expectation Maximization algorithm), as that particular introduction aims to bring a very high-level understanding of the topic without burdening the reader with additional prerequisite requirements. Therefore, in that particular sequence, the reader is first given a gentle introduction to the topic, then the necessary prerequisite (Maximum Likelihood) for understanding the more advanced introduction.

4.4.1 Error analysis
The extracted sequences are not without errors. These errors stem from several potential sources, as a fairly involved pipeline lies between the raw document and the resulting optimal sequence, providing an opportunity for errors to propagate through the different stages. We break down these errors by their source to give a better understanding of how these problems need to be addressed in future work:

**Terminology extraction:** The greatest source of errors stems from errors in terminology extraction. There are two types of errors involved in terminology extraction: false negatives (missing terms) and false positives (term sense disambiguation errors). False negatives are more difficult to detect and often result in missing prerequisites; missing terms are especially difficult when relying on a finite vocabulary.

**Explain/Assume classification:** The second greatest source of errors are the mistakes made by the aspect classifier. Classifying an explained term as an assumed term creates unnecessary prerequisites, while the reverse results in missing potentially important prerequisites.

**Path optimization:** because we solve the optimization problem exactly (i.e. find a global optimum), there are no errors stemming from the optimization itself (this will become a potential source of errors, however, when an approximation scheme, e.g. LP rounding, is used to obtain an approximate solution). However, the formulation of the optimization problem can be improved so as to introduce robustness to the errors in the earlier stages of the pipeline. As path optimization is the final stage that produces the final output, its sensitivity to the errors in terminology extraction and term aspect classification are directly reflected in the resulting output. Introducing robustness to these errors directly in the formulation of the optimization problem is potentially the most effective way to address the issues in the earlier stages of the pipeline. One issue with the current formulation is its inability to incorporate the relative frequency of the term into the optimization objective: ideally terms that appear less frequently in a document should have a lesser precedence for coverage than those that appear more frequently (Assumption 2 in the Introduction). The example in Figure 5 demonstrates the lack of robustness in the third document, where the appearance of the term integral creates an additional sequence of documents that cover this concept. From our earlier analysis is Section 3.1, we have shown that the errors in the Explain/Assume classifier are directly related to the relative frequency of the terms, and thus a way to incorporate these frequencies as weights into the optimization would potentially be the most effective way to deal with this noise.

5. CONCLUSION
We developed what we believe is the first end-to-end approach towards automatic curriculum extraction from the web, relying on the following pipeline: 1) extracting what is assumed vs. what is explained in a single document and then 2) connecting these documents into a sequence ensuring that the progression builds up the knowledge of the learner gradually towards their goal. We developed algorithms that addressed both of these components: 1) a semi-supervised approach for learning a term aspect classifier from a very small set of annotated examples and 2) an optimization problem for learning path recommendation based on the user’s learning goals. To the best of our knowledge, we for the first time demonstrate and leverage the most unique characteristic of the web in the domain of learning: diversity, i.e.
Figure 5: An example optimal sequence for the target document on Maximum Likelihood Estimation. Left: the term-coverage diagram. Each column represents a single web-page and each row a single term. Red rectangles correspond to terms that are classified as assumed in the corresponding document and blue corresponds to the explained terms. Right: the term-cover diagram is converted into a directed graph whereby an edge is drawn to a document from its closest prerequisite that explains at least one assumed term.

Figure 6: Additional examples of optimal paths generated from the 1,000-document web-page corpus for a select set of target web-pages. See text for details.

Acknowledgements
This research was funded by a grant from the John Templeton Foundation provided through the Metaknowledge Network at the University of Chicago. Computational resources were provided in part by grants from Amazon and Microsoft.
6. REFERENCES


ABSTRACT
Peer-grading is widely believed to be an inexpensive and scalable way to assess students in large classroom settings. In this paper, we propose calibrated self-grading as a more efficient alternative to peer grading. For self-grading, students assign themselves a grade that they think they deserve via an incentive-compatible mechanism that elicits maximally truthful judgements of performance. We show that the students’ self-evaluation scores obtained via this mechanism can be used to perform classic item response theory (IRT) analysis. In order to obtain unbiased estimates of the IRT parameters, we show that the self-assigned grades can be calibrated with a minimum amount of input from instructors or domain experts. We demonstrate the effectiveness of the proposed calibrated self-grading approach via simulations and experiments on Amazon’s Mechanical Turk.

Keywords
Assessment, self-grading, item response theory (IRT).

1. INTRODUCTION
A significant bottleneck in scaling traditional classrooms to hundreds or thousands of students is the challenge of enabling efficient mechanisms of assessment. Peer-grading, hailed as a solution to this “scaling problem,” has received significant attention, both from the education [12, 5] and machine learning [10, 11] communities. Broadly speaking, peer-grading can be thought of as a relaxation of the traditional teacher/student roles in the classroom: An expert instructor is replaced by several “noisy” students having the task of estimating performance of other students. Virtually all of the existing statistical models for peer-grading aim to estimate the student’s true performance from such noisy measurements, under some metric of optimality.

Self-grading constitutes a special case of peer-grading: The student is their own only “peer” and is solely responsible for assigning a score based on the judgement of their own work.

Depending on the student’s honesty in self-evaluation, self-grading is appealing for at least two reasons: (i) Students can provide a richer signal towards their internal state of knowledge by explicitly revealing confidence in their answers—a signal that can be exploited during assessment; (ii) because every student is their own grader, potentially no additional peer-grading efforts are required to perform assessment. Self-grading, however, introduces two unique challenges not faced in traditional peer-grading: (i) Designing mechanisms for eliciting honest judgement of performance and (ii) accounting for individual biases in self-evaluation. The first challenge in self-grading fundamentally requires an explicit mechanism for eliciting truthful judgements.\footnote{This is also a potential problem in peer-grading when conflicts of interest are present.} The second challenge is addressed in peer-grading by appealing to statistics and assuming that the population of graders is—at least on average—unbiased.

In this work, we propose calibrated self-assessment to address both of the above challenges. Our approach combines self-assessment with a small number of instructor-graded items, which provides a simple, incentive-compatible mechanism of eliciting self-assigned scores, and yields assessments of comparable or superior quality to a setting with significantly more instructor-graded items and no self-scoring. As a consequence, calibrated self-assessment enables a significant reduction in effort of instructors, domain experts, or peers.

2. RELATED WORK
We focus our review on two research directions that our work aims to bring together: (i) self-assessment as a method for summative assessment and (ii) decision-theoretic mechanism design for judgement elicitation.

Self-grading and Peer-grading in education: Self-assessment is often seen by teachers as a valuable tool in classrooms [17], who cite self-assessment as a viable way to reduce the instructor’s effort, elicit additional information from students (e.g., their effort and confidence), and provide an additional learning opportunity in the process. More recently, in addition to peer-grading, self-grading was deployed in massive open online courses (MOOCs) [5]. Self-grading as a tool for summative assessment, however, is controversial, with its validity questioned on the basis of students’ internal biases. In fact, studies indicate that bias is often a function of one’s ability [17, 16]. Studies that compare peer-grading and self-grading differ in their findings, with self-grading and
peer-grading performance excelling in different conditions (classrooms, age-groups, etc.), but both are heavily influenced by the underlying assessor biases (see [16] for a survey of the studies). A study carried out in four middle-school science classrooms found that peer-grading and self-grading have a high correlation with instructor grades, with grading bias patterns that are consistent with other studies [12]. In addition, they found that the process of self-grading resulted in learning gains, whereas peer-grading did not. A recent study carried out at the university level, however, found that both peer-grading and self-grading results in learning gains as a side-effect of grading [8].

The existing literature on self-grading points to the significant effect of bias in self-scoring, with most studies concluding that students of lower ability tend to inflate their grades more. As a consequence, we argue for the importance of an incentive-compatible mechanism that is designed to elicit maximally truthful judgements, and a calibrated model that is able to explicitly de-bias the individuals by incorporating a subset of instructor-graded items.

**Judgement elicitation:** The literature on truthful judgement elicitation through scoring functions dates back to the fifties, when the so-called “quadratic scoring rule” was proposed for the task of weather forecasting [2]. Since then, a number of generalizations of the quadratic scoring rule and other incentive-compatible scoring rules have been proposed and analyzed [3, 14, 7, 13] and found application in forecasting weather, sports, and finance. Analysis of the behavior of non-risk-neutral agents in scoring-rule-based mechanisms has received only limited attention [9], with lottery-based payoffs being the most well-known solution for encouraging risk-neutral behavior. Lottery-based payoffs had received mixed results in experimental evaluations [4, 15], and in the context of education a reward system based on a lottery is not a reasonable solution. In this work, we rely on heavily limited instructor input in order to correct for individual biases, which includes under- and over-confidence, as well as non-risk-neutral behavior.

To the best of our knowledge, the only work that applies a scoring rule mechanism in the context of education that we are aware of is [1]. The focus of this work is in analyzing the effect of different scoring functions on the self-assessment behavior of students. Our primary contribution in this work is in developing a principled statistical model for calibrated summative assessment that integrates self-scoring and instructor-scoring within the classic IRT framework.

3. **MODEL**

Self-grading without a proper incentive mechanism may lead to dishonest behavior. In the setting of self-grading, a “mechanism” is a scoring rule that specifies the rules by which the points are assigned to the student as a function of their own judgement and the outcome (i.e., whether their answer was correct). A mechanism is called incentive compatible when the student’s optimal strategy with respect to his or her own utility function results in a truthful elicitation of information, e.g., truthful judgement of their own work.

We consider the following scoring function:

\[
p_{ij} = \begin{cases} 
\theta_{ij} & \text{if correct} \\
-\frac{1}{2} \theta_{ij}^2 & \text{if wrong,}
\end{cases}
\]

where \(\theta_{ij} \in [0, A]\) is a score provided by student \(i\) in answering question \(j\), where \(A\) is some fixed upper bound. If the student provides a correct answer, they get the \(\theta_{ij}\) points that they proposed; if they provide an incorrect answer, they lose exactly half of that value squared. This scoring function is known as a quadratic scoring rule and was first proposed in [2].

For this scoring function, the expected payoff is

\[
E[p_{ij}] = \theta_{ij} \hat{\pi}_{ij} - \frac{1}{2} \theta_{ij}^2 (1 - \hat{\pi}_{ij}),
\]

where \(\hat{\pi}_{ij}\) is the \(i^{th}\) student’s estimate of the probability that they will get question \(j\) correct. This expression is maximized when

\[
\theta_{ij} = \frac{\hat{\pi}_{ij}}{1 - \hat{\pi}_{ij}}.
\]

Equation 2 is exactly the student’s own belief about the odds of them answering the question correctly. Consider that the student estimates their chances of answering any question correctly by simultaneously estimating their own ability and the difficulty of the question. Let us now define that probability to be the standard IRT Rasch likelihood, but defined with respect to the student’s own estimate of their ability, \(\hat{s}_i\), and their estimate of the question’s difficulty \(\hat{q}_j\):

\[
\hat{\pi}_{ij} = \frac{1}{1 + \exp(- (\hat{s}_i - \hat{q}_j))}.
\]

Given the student’s estimate of their own ability \(\hat{s}_i\) and of the difficulty of the question \(\hat{q}_j\), we can now derive their optimal proposed score (assuming they act rationally and are risk-neutral) for the problem \(\theta_{ij}\) (or rather its logarithm):

\[
\log(\theta_{ij}) = \hat{s}_i - \hat{q}_j,
\]

which follows from the fact that log-odds of a logistic model is a linear function of its parameters. We will assume that the student is risk-neutral and is unbiased in his or her estimates of own ability and question difficulty, but we will relax both assumptions later. On any given question, however, the student’s estimate of their ability to answer that particular question may deviate from their true ability. Assuming that the student’s own estimates are normally distributed around their true values, we get:

\[
\hat{s}_i - \hat{q}_j \sim \mathcal{N}(s_i - q_j, \sigma^2),
\]

where \(s_i\) and \(q_j\) are the true student ability and question difficulty parameters respectively. As a consequence, it follows that \(\log(\theta_{ij})\) is normal distributed and \(\theta_{ij}\) is log-normal distributed. Consider a dataset \(D\) consisting of the self-assigned scores \(\log(\theta_{ij})\) submitted by each student for each question that the student answered. We can write the conditional likelihood of the entire dataset as follows:

\[
P(\theta | s, q) = \prod_{(i,j) \in D} \mathcal{N}(\log(\theta_{ij}) | \mu = s_i - q_j, \sigma^2).
\]

Here, \(s\) and \(q\) are the vectors comprising the student ability and question difficulty parameters, respectively, and \(\theta\) is the vector of student-submitted scores. Maximizing the
likelihood of all observations gives a straightforward least-squares solution for the parameters $s_i$ and $q_j$, given all the user-provided scores $\theta_{ij}$. Note that $\sigma^2$ is assumed to be a constant variance in students’ estimates of their own ability. In practice this variance is likely user-specific and corresponds to the students’ ability in self-assessment. We will address the issues of bias and variance in self-assessment in Section 3.2.

### 3.1 Parameter estimation

It is interesting to note that we can solve for the IRT parameters (student abilities and question difficulties) using the above formulation with no outcome information, i.e., without knowing which students answered which questions correctly. In fact, the above approach does not even require that the students who are self-grading know what the correct answer is; students’ confidence in their answers elicited through the quadratic scoring rule is all that is needed to learn the parameters of the model. Of course, this observations relies on two fundamental assumptions: (i) students are risk-neutral and (ii) students are unbiased in estimating their chance of answering a question correctly. In Section 3.2, we will account for the individual biases and non-risk-neutral behavior by explicitly introducing a bias parameter into the model and estimating it from an additional set of instructor-graded responses. However, in order to gain a better understanding of the model, it is insightful to first analyze the solution to the problem where both of these assumptions hold.

The solution for the model parameters can be obtained in closed-form using a standard pseudo-inverse solution to a least-squares problem. Alternatively, the solution can be obtained iteratively, without requiring to explicitly invert any (potentially large) matrices. In particular, one can repeatedly evaluate the following two steps:

$$s_i = \sum_{j \in Q_i} \frac{q_j}{\lambda + n^q_j} + \sum_{j \in Q_i} \log(\theta_{ij}) \left( \frac{1}{\lambda + n^s_j} \right)$$

$$q_j = \sum_{i \in S_j} \frac{s_i}{\lambda + n^s_i} - \sum_{i \in S_j} \log(\theta_{ij}) \left( \frac{1}{\lambda + n^q_i} \right).$$

Here, $s_i$ is the ability of student $i$ and $q_j$ is the difficulty of question $j$. To guarantee a unique solution, we introduce a non-negative regularization parameter $\lambda$, which we will discuss in more detail in the next paragraph. The constants $n^q_j$ and $n^s_i$ are the number of questions that student $i$ answered and the number of students that answered question $j$ respectively. Note that the above iterative solution has an intuitive interpretation: The ability of the student is the sum of the average of the (log-transformed) self-assigned scores to a set of questions that the student answered and the average difficulty of those questions. In turn, the difficulty of a question is the negative of the average (log-transformed) score that students assigned to themselves for that question plus the average ability of the students who answered that question. Intuitively, if students with high ability self-assess themselves to have done poorly on a specific question, that question will have a large difficulty parameter.

In the case where there is no missing data, i.e., each student answers each question, the solution for student ability parameters simplifies to:

$$s_i = \left[ \frac{\sum_{j \in S_i} \log(\theta_{ij})}{\lambda + n^s_i} \right] + \mathcal{O}(1/\lambda) \mathbf{1},$$

where $\mathcal{O}(1/\lambda)$ is a function that grows proportional to $1/\lambda$. In other words, the student’s ability is simply the average of the (log-transformed) scores that the student assigned to themselves plus a constant that is identical for each student. This solution also illustrates the role of the regularization parameter $\lambda$. Because the solution for $s$ and $q$ is location-invariant, without an explicit prior, the likelihood is maximized by scaling all parameters to infinity. This is equivalent to setting $\lambda$ to 0, in which case the above solution will tend to infinity, as expected. Note, however, that the relative ranking of the student abilities in this solution will be consistent, regardless of $\lambda$. As obtaining the ranking of the students is our primary focus, we can thus set $\lambda$ to zero in the above solution, and simply consider the average self-assigned (log-transformed) score as the ability parameter of the student. The same argument applies to question difficulty parameters.

### 3.2 Calibrating the model

There are two issues in relying on students’ self-assigned score for ranking students via the IRT model: (i) Students may be prone to over- or under-estimating their ability and (ii) because there is uncertainty involved in both answering and grading, some students may be more or less inclined to “gamble” with their self-assigned score (i.e., some students are more or less risk-averse/risk-loving). We subsume both effects as it is impossible to tell them apart into a general student “bias” in self-grading, and model it explicitly as

$$\log(\theta_{ij}) = s_i - q_j + b_i,$$

where $b_i \in (-\infty, \infty)$ is a student-specific bias. We assume that this student bias is drawn from a normal distribution $b_i \sim \mathcal{N}(0, \sigma^b)^2$, where the above distribution stipulates that the average of the student population is unbiased. It is impossible to estimate $b_i$ using self-grading alone, as without actual observations of correctness of students’ responses, the model will confound $s_i$ and $b_i$ into a single parameter. Imagine that we do grade a student’s responses on a small subset of the answered questions (which they also self-grade). Let the set of instructor-graded questions be $Q_L \subseteq Q$, where $Q$ is the set of all questions. As the observations of instructor- and self-assigned grades are all conditionally independent given the student and question parameters, the overall likelihood of both self- and instructor-graded scores is a product of these likelihoods. We can then express the log-likelihood of the entire dataset as a sum of the self-graded response log-likelihoods and instructor-graded response log-likelihoods:

$$\log P(\theta, y | s, q, b) = \sum_{s \in S} \left( \sum_{q_j \in Q} \log(\theta_{ij}) - (s_i + b_i - q_j) \right)^2$$

$$+ \sum_{q_j \in Q_L} \log(1 + \exp(-y_{ij}(s_i - q_j))) \right).$$

Here, $s_i$ is the ability of student $i$ and $q_j$ is the difficulty of question $j$. To guarantee a unique solution, we introduce a non-negative regularization parameter $\lambda$, which we will discuss in more detail in the next paragraph. The constants $n^q_j$ and $n^s_i$ are the number of questions that student $i$ answered and the number of students that answered question $j$ respectively. Note that the above iterative solution has an intuitive interpretation: The ability of the student is the sum of the average of the (log-transformed) self-assigned scores to a set of questions that the student answered and the average difficulty of those questions. In turn, the difficulty of a question is the negative of the average (log-transformed) score that students assigned to themselves for that question plus the average ability of the students who answered that question. Intuitively, if students with high ability self-assess themselves to have done poorly on a specific question, that question will have a large difficulty parameter.
Here, \( y_{ij} \in \{-1, 1\} \) is the instructor-grade for question \( j \) answered by student \( i \) and \( y \) is the response vector for all students \( (y_{ij} = +1 \) corresponds to a correct response and \( y_{ij} = -1 \) otherwise). Observe that the “bias” parameter only appears in the self-graded part of the likelihood. This allows us to calibrate the model via instructor-graded questions as a “training set” to separate the effects of the bias and true ability. Note that, unlike in the previous case that relied entirely on students’ self-scores, like with the traditional Rasch IRT model, we are unaware of a closed form solution for this formulation. In all of our experiments, we use the L-BFGS algorithm [18] for learning model parameters.

3.3 Consequences of students’ awareness of the mechanism

The assumption that the learner is optimizing a utility function based on the expected test score:

\[
\mathbb{E}[p_{ij}] = \theta_{ij} \hat{\pi}_{ij} - \frac{1}{2} \theta_{ij}^2 (1 - \hat{\pi}_{ij})
\]

fundamentally assumes that the student believes that each question will be graded, as otherwise there would be no possibility of getting a question wrong and losing points. In practice, our goal for self-grading may be motivated by the effort to reduce the instructor’s involvement in grading, and, in general, as a way to scale assessment to potentially very large classrooms, such as massive open online courses (MOOCs). Having each submission be graded by an instructor (or your peers) defeats the purpose of self-grading. If, however, the student is aware of the fact that not every question is graded, we can expect that their utility function, and thus their optimal strategy, will be affected by this knowledge. If the test is administered once, of course, the students could be deceived into believing that every question is graded. In a real course, however, a more realistic assumption is that the students possess the knowledge that not all of the questions are graded and if the assignments are returned, we can expect that the students’ estimates of the fraction of graded questions will improve over time. If, however, the student believes that a random subset of their submissions is graded by someone else, but if the student does not know which subset is graded, then we should still expect the student’s optimal behavior to be maximizing a utility function similar to the one above. The utility function will not be the same, as we now have to account for the student’s belief about how many problems are graded by someone else. Let us assume that the student has a prior belief that each problem has a probability \( \rho \) of being graded. Then, the expected score the student \( i \) receives on question \( j \) is given by

\[
\mathbb{E}_{\rho}[E[p_{ij} \mid \text{graded}]] = \rho \theta_{ij} \hat{\pi}_{ij} - \frac{1}{2} \theta_{ij}^2 (1 - \hat{\pi}_{ij}) + (1 - \rho) \theta_{ij},
\]

where we take an additional expectation with respect to the student’s belief that the problem is graded. Note that when a problem is not graded, the expected score that the student receives is just \( \theta_{ij} \), i.e., their self-assigned score, regardless of whether the student answers correctly. This is because when a problem is not graded, there is no possibility of losing points. We can show that the student’s optimal self-assigned score \( \log(\theta_{ij}) \) has the following approximate relationship to their ability and question difficulty (the approximation is a piece-wise linear approximation to the true strategy that is asymptotically accurate):

\[
\log(\theta_{ij}) = \max \left\{ \log \left( \frac{1}{\rho} - 1 \right), (s_{ij} - q_{ij}) - \log \rho \right\}.
\]

The optimal strategies for different values of \( \rho \) are illustrated in Figure 1. The student’s knowledge of the mechanism is reflected by the appearance of a lower-bound on the self-assigned score in a region where the student is likely to do poorly (low values of \( s_{ij} - q_{ij} \)). This is expected: If the student is aware that the chance of a particular question to be graded is low enough, it would make sense to take advantage of those odds and “bet” a small, but a non-zero amount, even if the student does not know the correct answer. From a practical perspective of implementing a system that solicits self-assessment scores, it would not make sense to provide the user with the ability to provide a self-assessment score lower than their optimum. From the model inference perspective, this introduces a complication: Observations that correspond to the lowest possible self-score do not correspond to any specific \( s_{ij} - q_{ij} \), but rather an entire range. This problem is known generally as censored regression, and can be solved using the same approach as for the original problem, but with the modified likelihood function that accounts for this “kink.” Note that a similar restriction on the likelihood (but as an upper-bound) is introduced when the maximum attainable score for a problem is incorporated into the scoring function.

4. EXPERIMENTS

4.1 Simulations

It is insightful to study the effect of biased grading in the population of students on the quality of the learned parameters in the IRT model: student ability parameters and question difficulty parameters. We perform a simple simulation of a classroom with 50 questions and 30 students (question difficulties and student abilities are sampled from a zero-mean normal distribution with a standard deviation of 3), where each student answers each question (a total of 1,500 responses). In this simulation, each student submits their self-grade \( \log(\theta_{ij}) \) for
Student rank correlation

Figure 2: Simulation results. Rank correlation across students obtained using three models for different variance of self-grading bias ($\sigma_b^2$): (i) black: a model that uses student self-scores and the correctness of their response to a subset of graded questions (number of graded questions on x-axis), (ii) solid gray: a model that uses correctness of their response to a subset of graded questions only (number of graded questions on x-axis) and (iii) dashed gray: a model that uses only the students’ self-score.

Each question by optimizing their utility according to the utility function in 3. We repeat the simulation for four different populations of students, each with a different variance $\sigma_b^2$ of the bias parameter. To evaluate the quality of the inferred student parameters, we compute the rank correlation (Kendall Tau) between the true ordering of the students (by their true parameters) and the ordering obtained by sorting the students based on the inferred parameters. The Kendall Tau metric is defined as follows:

$$KendallTau(s, \hat{s}) = \frac{N_{\text{correct}} - N_{\text{wrong}}}{N_{\text{pairs}}}$$

where $s$ and $\hat{s}$ are the true and inferred student ability parameters, respectively, and $N_{\text{correct}}$ and $N_{\text{wrong}}$ is the number of student pairs that are ordered correctly in the inferred ranking (with respect to the true ranking) and the number of pairs that are ordered incorrectly, respectively. Kendall Tau is equal to +1 when the rankings are consistent and to −1 when the rankings are inverted. The corresponding results are shown in Figure 2.

Three models were evaluated:

- **Self-grading only**: Only students’ self-submitted scores $\log(\theta_{ij})$ are used in fitting the Rasch model parameters.

- **Instructor-grading only**: Only the correctness of the responses is used for fitting the Rasch model parameters; this is a classic Rasch model. We vary the number of questions used in fitting the model parameters (x-axis in Figure 2).

- **Self-grading + instructor-grading**: A combination of self-scores submitted by all students for all questions and the correctness of a subset of submitted questions is used for fitting the Rasch model parameters (number of questions used is the x-axis in Figure 2).

In the case where the students in the class are relatively unbiased (low $\sigma_b^2$) (top left in Figure 2), self-scoring achieves a better rank-correlation than the traditional IRT Rasch model, even when many questions are instructor-scored. Interestingly, in the regime of low bias, including actual instructor-graded responses actually negatively affects the correlation (this is due to over-fitting caused by a small number of instructor grades—introducing additional bias variables requires a sufficient number of observations to infer them reliably; this performance drop eventually disappears when a sufficient number of observations to infer them reliably; this performance drop eventually disappears when a sufficient
number of questions is included). As the bias of the population increases, the performance of the self-scoring model decreases but still exceeds the performance of the instructor-only Rasch IRT, especially in situations where only a few questions are scored.

4.2 User study
To evaluate the efficacy of the proposed self-grading approach, we conducted a user-study on Amazon’s Mechanical Turk. We solicited 206 subjects to participate in a task titled “Do a short math quiz and earn bonus!”. The subjects were asked to answer 30 math questions of varying difficulty levels ranging from basic arithmetic to pre-calculus. The questions from the dataset introduced by [6] were used in our experiment. All questions were multiple choice and included a “none of the above” option, included in order to minimize the probability of getting a right answer through a process of elimination. Although in practice, multiple-choice questions mostly defeat the purpose of self-grading, we use multiple choice questions for the ease of evaluation and the lack of subjectivity that would be otherwise present in free-response questions. Figure 3 illustrates a single question from the task. The subjects were asked to mark what they believed to be the correct answer, and then to assign themselves the number of points that they would receive if they answered the question correctly. The input was provided through a slider. Moving the slider automatically displayed the number of points that the subject would gain if they answered the question correctly (green) and the number of points they will lose if they answer the question incorrectly (red).

We follow the same evaluation scheme that we described in the previous section: (i) vary the number of instructor-graded questions from 0 to all questions (30) and combine that with the self-assigned scores for every question, (ii) infer the ranking using the proposed model, and (iii) compare it to the ranking that is derived from “gold-standard” proxy.

We find that the results are comparable to those obtained in the simulation (Figure 4(a)). Self-scoring is already able to obtain a reasonable correlation with the “gold-standard” ranking even without any instructor-graded question. Incorporating instructor-grades for additional questions improves the performance. Rank correlation metrics, such as Kendall Tau, while convenient for summarizing the results with a single quantity, often fail to distinguish regimes where the model might perform differently. It is instructive to consider the performance of rank-correlation in the different segments of the ranking. Figure 4(b) decomposes the results by quartiles. We employ a more intuitive metric, Precision@ Quartile, defined as follows:

$$\text{Precision} @ Q_i = \frac{|S_{Q_i} \cap S_{Q_i}|}{|S_{Q_i}|}$$

where $S_{Q_i}$ is the set of students in the $i$th quartile of the “gold-standard” ranking, and $S_{Q_j}$ is the set of students in the $j$th quartile of the inferred ranking. This metric captures the ability of the model to perform within a particular segment of the ranking. For example, looking at Precision at the first quartile, measures the ability of the model to predict top students. From Figure 4(b) we can conclude that the model is significantly better at distinguishing the top-ranked students (first quartile) as compared to the lower-ranked students (second quartile). By using the self-scoring signal without any instructor-graded questions, we are able to recover nearly 60% of the top quarter of all students. The performance in the second quartile is significantly lower, but follows the same trend: incorporating the students’ self-reported scores in the regime of zero to several questions significantly improves performance over the baseline of instructor-graded questions alone. This observation leads to the conclusion that, at least in this study, better students were better at estimating their ability. We look into the effect of self-estimation performance in more detail in the next section.

4.3 Self-assessment and bias
The performance of the model that relies on self-assessment depends fundamentally on the model’s estimates of the students’ biases as well as the ability of the students to self-assess reliably (self-assessment variance). In our model, we infer only the individuals’ biases and assume constant variance in the self-assessment likelihood (these could in principle be estimated as well). Figure 6 illustrates the individual inferred biases for each student (averaged across multiple folds), sorted in an increasing order. The resulting distribution illustrates the skew in the bias distribution towards “under-confidence,” i.e., most students tend to under-estimate their ability (act conservatively). The importance of estimating bias is underlined in Figure 4(a), where we include an additional baseline Self-Scored + Graded (no bias) (light solid line). This baseline combines self-assessment and instructor-grades but does not incorporate the explicit student-bias parameter. As
Figure 4: User study results. Rank correlation across students obtained using three different models (i) **Self-scored**: a model that relies entirely on student-submitted self-assessments, (ii) **Graded**: a model that relies entirely on instructor-provided grades, as a function of the number of graded questions (x-axis), and (iii) **Self-scored + Graded**: a model that aggregates students’ self-assessment scores on all questions and a variable number of instructor-graded questions (x-axis). (a) Computes rank correlation across all students using Kendall Tau, and (b) decomposes rank correlation across the first two quartiles using the *Precision@Quartile* metric. The model that combines self- and instructor-assigned scores is significantly better at predicting the top-performing students (first quartile). Combining instructor grades with self-assessment significantly improves both rank measures, especially when only a few questions are graded. Note that the total number of questions in the study was 30; we display the results up to 15, as the differences between both models is not substantial beyond that.

Figure 5: Bias vs. ability (centered). Both parameters were inferred using all of the available data. Each point in the scatter-plot corresponds to one student. A weak, but significant correlation between bias and ability exists.

Figure 6: Inferred bias parameter of each student (sorted in an increasing order). The bias parameter was inferred using all of the available data.

It is potentially insightful to investigate the relationship between self-assessment bias and ability. We consider the inferred bias parameter after incorporating instructor-grades for all questions, and compare it to the inferred ability parameter of each student. The result is illustrated in the scatter-plot in Figure 5. While the relationship between the two is not strong, there exists a negative correlation between ability and self-assessment bias (Pearson’s correlation: 0.17, p-value = 0.013). Students that are more able tend to underestimate their ability, and students that are less able tend to inflate their ability. This finding is consistent with the literature in self-assessment [17, 16].

5. **CONCLUSION AND FUTURE WORK**

In this work, we have developed a novel approach for performing calibrated, summative self-assessment by combining (i) student’s self-evaluations obtained via an incentive-compatible scoring mechanism and (ii) a minimal number of instructor-graded responses. We have shown that when the scoring rule is quadratic, the standard IRT Rasch model reduces to standard linear regression. We have demonstrated that the quality of the inferred assessment using self-scoring alone without additional instructor input is, on-average, comparable to the performance obtained using the standard IRT that requires significant instructor effort. Furthermore, by incorporating a minimum number of instructor-graded responses, we have shown that our approach substantially improves the estimates of the students’ abilities and the
questions’ difficulties. Finally, we have addressed the longstanding issue of applying scoring rules in practice: dealing with the consequences of individuals’ biases and non-risk-neutrality. We have proposed to explicitly model the combined effect of these two factors within the standard IRT framework, allowing the model to effectively de-bias these individual differences.

Our results open an interesting direction of inquiry: are there other scoring functions that are more efficient at estimating IRT parameters, and if so, can the scoring functions be adapted to individual students and questions, improving the efficiency of adaptive testing? In order to facilitate further research in this direction, we release all code and data used in this study.

6. ACKNOWLEDGMENTS

We would like to thank Mr. Lan, Ph.D., as well as A. E. Waters and R. G. Baraniuk for their help with the Mechanical Turk dataset [6]. The work of I. Labutov was supported in part by a grant from the John Templeton Foundation provided through the Metaknowledge Network at the University of Chicago. The work of C. Studer was supported in part by Xilinx Inc. and by the US NSF under grants ECCS-1408006 and CCF-1535897.

7. REFERENCES

ABSTRACT
The advent of Massive Online Open Courses (MOOCs) has led to the availability of large educational datasets collected from diverse international audiences. Little work has been done on the impact of cultural and geographic factors on student performance in MOOCs. In this paper, we analyze national and cultural differences in students’ performance in a large-scale MOOC. We situate our analysis in the context of existing theoretical frameworks for cultural analysis. We focus on three dimensions of learner behavior: course activity profiles; quiz activity profiles; and most connected forum peer or best friends. We conclude that countries or associated cultural clusters are associated with differences in all three dimensions. These findings stress the need for more research on the internationalization in online education and greater intercultural awareness among MOOC designers.

1. INTRODUCTION
Over the past decade there has been a substantial increase in the study of cross-cultural behaviors in e-learning systems. Prior researchers have shown that learners from different cultures behave differently when using educational systems, particularly in terms of their off-task behaviors [25, 21], help-seeking [21], and collaboration [22, 16]. The cultural differences uncovered in these studies suggest that designers of future e-learning platforms would benefit from a better understanding of their distinct target populations and distinct cultures.

Large-scale MOOCs typically attract diverse international audiences. The course we discuss here, for example, attracted students from 172 countries on 5 continents. Despite this acknowledged diversity, most MOOCs take a one-size fits-all approach to designing and structuring the course. The materials are typically offered in a single format and language, or via direct translations that preserve the structure, pacing, and content.

Prior researchers have shown that country of origin affects students’ performance in MOOCs. Nesterko et al. [19] found that non-American students were more prone to complete MOOCs and to seek certification than their U.S. counterparts. Guo and Reinecke [12] found that a student’s country of origin significantly predicted the amount of content that they would cover and the amount of time that they spent reviewing prior course content. Kizilcec [17] found that there was a significant correlation between a country’s level on the Human Development Index and the number of students from that country who completed a majority of the assignments. In each of these studies, however, nationality was treated as a single independent factor. No substantive comparisons were made between countries or cultures, nor did the authors frame their conclusions in the context of prior theoretical work on cultural differences in learning.

A deeper understanding of how students differ both within and across cultures will help us to design and deploy more effective, and truly international MOOCs. And this understanding will be enriched by relating these differences to the rich existing literature on cross-cultural education such as Hofstede’s cultural dimensions theory [13] and the Cultural Dimensions of Learning Framework (CDLF). In this paper we will address this need through our analysis of crosscultural student behaviors in an existing MOOC. This was an open course with a total enrollment of 29,149 students drawn from 172 countries and 5 continents. We found clear inter-country and inter-cultural differences in the observed stu-
2. LITERATURE REVIEW

2.1 Culture & Educational Technology

Advances in educational technology have enabled educators to incorporate technologies at larger scale and to collect richer and more diverse educational data than ever before. This has, in turn, substantially increased interest in studying variations in the use of e-learning tools across cultures.

One approach to understanding the impact of culture on learning is through field observation. Rodrigo et al. [25] coded U.S. and Filipino students’ on- and off-task behaviors when using three ITSs. They found that Filipino students spent more time on task than their U.S. counterparts on all three systems. They also found that the Filipino students engaged in some systems more than others. Similarly, Ogan et al. [22] coded the on-task behaviors and interaction of similar students in Chile. They found that the Chilean students had a higher proportion of on-task interactions than the U.S. students studied previously.

Another approach is through educational data mining. Ogan et al. [21] generated student models from ITS logs collected in three countries: Costa Rica, The Philippines, and the U.S. Their goal in this work was to predict effective help-seeking behaviors. They found that it was possible to generalize the U.S. model to Filipino students but not to students from Costa Rica. Saarela and Karkkainen [27] applied a hierarchical clustering algorithm to data collected from the PISA, a worldwide assessment of 15-year old students covering reading, mathematics, and science. They found that students’ performance on the test clustered by country, suggesting cultural influences.

While these studies found interesting cross-cultural differences, we have little understanding of why these differences occur, or of how they relate to more general cross-cultural variation. Learning behaviors are influenced by a complex set of factors and cross-cultural comparisons may help us deepen our understanding of this phenomenon and highlight ways to remediate or accommodate it. In this paper, we explore the logs of student activity in a MOOC, with an eye toward how culture may relate to differences in behavior.

2.2 MOOC Research

MOOCs represent both opportunities and challenges for educators. On the one hand they involve large numbers of users working in highly instrumented systems which can, in turn, provide deep insights. On the other hand, however, MOOCs have high dropout rates, wide variation in levels of engagement, and MOOC users have extremely diverse motivations and demographic backgrounds. Thus any insights are qualified by the noisy nature of the data. Researchers have therefore focused their efforts on better understanding MOOC users and their differing behavior patterns.

One approach to understanding MOOC students is to build predictive behavior models based upon their clickstream data, such as mining sequences of actions for analysis [29, 5]. These induced models are highly accurate but are not always readily interpretable. Other work has focused on improving our understanding of engagement and dropouts by detecting key subgroups. In this work, researchers have used hierarchical clustering to identify groups of students with similar patterns of engagement, such as those who viewed many lectures but rarely attempted quizzes, and those who balanced their activities equally [17, 10, 4, 1]. Kizilcec et al. [17] and Ferguson et al. [10], for example, clustered students by engagement factors such as the number of lectures viewed and quizzes attempted. Anderson et al. [3] likewise used lecture views and considered the ratio of lectures to assignments while Bergner et al. [4] focuses solely on assignments attempted. These studies served to highlight the distinct behavioral patterns of different subgroups.

Researchers have also begun to study students’ diverse backgrounds through voluntary surveys with the goal of understanding how their incoming motivation [28, 2] and demographic features [19, 17, 12] affect their observed behaviors. Both Nesterko [19] and Deboer [9] found that participation (as indicated by survey responses) and certificate attainment rates differed across countries, continents, and genders; they did not, however, delve deeper into students’ in-system behaviors as logged by the learning environment. Wang and Baker [28], by contrast, found that learners receiving course certificates tended to be more interested in course content, while students not receiving certificates often stated that they were seeking a new type of learning experience.

Few of these researchers however, have focused on the relationship between geographic information and observed behaviors. Guo and Reinecke [12] applied linear regression to correlate some demographic features such as years of education to geographic data. They found that a students’ country of origin was significantly related to their coverage of the course content overall and the extent to which they reviewed prior content, called backjumps. They attributed this diversity to varying student-to-teacher ratios. They found that countries with a higher ratio had a higher frequency of backjumps suggesting more time on review. In related work, Kizilcec focused on partitioning countries into tiers based upon the Human Development Index (HDI). They found that as the HDI tier increased, so did the proportion of students who completed the course. While these results are instructive, however, the authors made no attempt to situate these results in the context of existing theoretical models of cross-cultural learning.

Thus the results from prior MOOC research show that understanding students’ diverse backgrounds can be essential to the development of effective educational interventions, and to providing useful support for student engagement and participation. Geographical location, considered as a set of economic, cultural, and educational differences, may play a crucial role in understanding, supporting, and appealing to the increasing population of MOOC users.

2.3 Theoretical Frameworks

MOOCs and educational technologies allow us to collect robust information about cross-cultural differences in user behaviors. Yet we face challenges in interpreting and explain-
ing these results in a consistent theoretical framework.

Prior educational researchers have worked to identify related cultural dimensions and values, and to examine how they vary across cultures. One common framework is Hofstede’s Cultural Dimensions Theory [13, 14]. Hofstede analyzed a set of 117,000 attitude surveys collected by IBM from their international workforce and synthesized a set of 7 general cultural dimensions: a) power distance; b) collectivism vs. individualism; c) femininity vs. masculinity; d) uncertainty avoidance; e) long/short term orientation; and f) indulgence vs. restraint. Hofstede then calculated scores for each culture within these dimensions.

Hofstede’s dimensions have been used to analyze and explain differences in collaboration across cultures [16], as well as differences in help-seeking and off-task behavior in educational technology [21, 25]. However these studies have suggested that the cultural dimensions framework has some limitations in explaining these findings. Many of the key differences in the observed behaviors do not correspond to the differences that Hofstede’s theory suggests. In particular, variations in collectivism and collaboration/help-seeking strategies do not seem to relate well to Hofstede’s underlying dimensions. Therefore we will combine this with the Cultural Dimensions Learning Framework (CDLF).

The CDLF framework, designed by Parrish et al. in 2010 [24], defines eight cultural parameters regarding social relationships, epistemological beliefs, and temporal perceptions, and how they manifest in learning situations. The CDLF has been used to guide the design and analysis of e-learning across cultures [23, 15]. For the purposes of our analysis we will focus on the intersection of the CDLF and the Hofstede dimensions. We will use this hybrid framework to group countries into cultural clusters, and to interpret the observed behavioral differences between them. Table 1 provides an overview of the shared dimensions.

While these frameworks may help to explain observed behaviors, it is worth noting that learner behaviors in MOOCs can be affected by many other factors such as personal motivation. Wang and Baker [28], for example, surveyed the motivations of incoming students on a later version of the course we study here and found that learners who obtained course certificates tended to be more interested in course content than those who took the MOOC in order to test the learning experience. While this highlights the importance of individual differences, our analysis below will focus on inter-country differences and cultural factors.

### Table 1: Intersection of Hofstede Dimensions and the Cultural Dimensions of Learning Framework.

<table>
<thead>
<tr>
<th>Hofstede Dimension</th>
<th>Selected Interpretations in CDLF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Distance: the extent to which the less powerful members expect and accept unequal/unfair situations.</td>
<td>Countries with high power distance view teacher as an unchallenged authority and the primary communicator, not as a fallible peer.</td>
</tr>
<tr>
<td>Individualism: the degree of interdependence a society maintains among its members</td>
<td>Highly individualist students are more prone to speak up in class, to value diverse opinions in learning, and to be motivated by personal gain.</td>
</tr>
<tr>
<td>Masculinity: the degree to which a culture is motivated by competition (instead of life quality)</td>
<td>More masculine cultures are associated with increased levels of competition and a heavier pursuit of recognition.</td>
</tr>
<tr>
<td>Uncertainty Avoidance: The extent to which a culture feels threatened by ambiguous or unknown situations and tries to avoid these</td>
<td>Students who avoid uncertainty tend to focus more on getting the right answer from authoritative sources and from the structured learning activities.</td>
</tr>
</tbody>
</table>

3. DATA

The data used in this study was collected from Big Data in Education (BDE), an 8-week long MOOC offered by the Teacher’s College at Columbia University on the Coursera platform [28]. The BDE curriculum included video lectures, discussion forums, and 8 weekly assignments or quizzes. The lectures covered key methods for educational data analysis. The assignments required students to analyze existing data (typically real data collected from educational settings) and to answer questions about their results. All of the assignments were automatically graded via numeric or multiple-choice questions. Students were given between 3 and 5 attempts to complete each assignment with the best score being counted. Students were required to complete their assignments within 2 weeks of it being released. In order to obtain a certificate students were required to obtain an average grade of ≥70% over all 8 assignments. High performing students could receive a certificate with distinction. 638 students completed the course and obtained a certificate.

Data from this course has been previously used to study motivation [28], negativity [7], student communities [6], the relationship between linguistic quality of forum posts and completion [8], as well as longitudinal behavior patterns [31].

For the purposes of our analysis we analyzed clickstream data containing user IDs, IP addresses, URLs and timestamps for 29,149 students. This data included all 638 students who received a certificate as well as 750 who posted on the forum. After classifying students by behavior type we found that a total of 1,591 students were actively engaged with the course while the remaining 27,588 were ‘bystanders’ who enrolled but did not do any significant work. We assigned users to regions based upon their most frequent IP address as has been done in prior work [17, 9, 12]. The top 15 countries by registration are shown in Figure 1.

We then analyzed the URLs located in the clickstream data to identify the following major activities: view lecture (VL), attempt or submit quiz (AQ, SQ), and read or make a post in forum (RP, MP). We then generated activity sequences from this data using an n-gram approach consistent with prior research [29, 5]. Note that this data does not contain information about how long the student spent viewing a URL. The data only records individual mouseclicks. Therefore it functions as a record of student access but not...
4. METHODS AND RESULTS

We hypothesize that students from different countries or cultures will behave differently in the course. We chose to examine four research questions: RQ1. (Course Activity Profiles, CAPs) What are the primary categories of students based upon the frequency (both total and relative) with which they accessed different course activities? RQ2. (CAPs by Country) Does the proportion of student categories differ by country? RQ3. (Quiz Activity Profiles, QAPs) When do students in each category access the different types of course activities and how is that correlated with quiz submissions? RQ4. (QAPs by Culture & Country) How do quiz-based activity profiles and countries relate to the four overlapping Hofstede/CDLF cultural dimensions of: power distance, individualism, masculinity, and uncertainty avoidance? RQ5. (Forum best friends) Is a student’s most frequent forum partner in the same country/culture?

For RQ1, we used hierarchical clustering to identify five course activity profiles (CAPs) (e.g. students who focused solely on quizzes). For RQ2, we clustered countries by the proportion of students who fit each CAP in order to determine whether or not students from a given country are more likely to fit one CAP over another. For RQ3, we partitioned the course data by quizzes and examined whether or when students in each CAP accessed the lectures, quizzes, and forum content. This led to the development of Quiz Activity Profiles (QAPs). For RQ4, we then clustered students based upon their cultural dimensions and compared the QAPs by culture and student category (CAP). For RQ5, we performed a χ² analysis to investigate whether the students’ most frequent interlocutor on the forums were more likely to be drawn from the same country/culture. In each section below, we will present the methods and results for each of these questions in greater detail.

4.1 RQ1: Course Activity Profiles, CAPs

What are the primary categories of students based upon the frequency (both total and relative) with which they accessed different course activities? Prior researchers have used hierarchical clustering to discover meaningful subgroups such as: users who viewed many lectures but rarely attempted quizzes and users who balanced the number of lectures viewed and quizzes attempted [17, 10, 4, 1].

In this work we applied hierarchical clustering to classify students based upon the proportion of activities that they engaged in over the course. These included: lectures accessed, quizzes attempted, and form posts made or accessed. We found that clustering students by the the number of lectures that they accessed and quizzes attempted yielded five interpretable clusters which we designated solvers (generally take more quizzes), viewers (generally watch more lectures), all-rounders (balance both), samplers (watch some lectures and do a quiz), and bystanders (do very little). Table 2 shows the CAP clusters with average silhouette widths (ASWs) in excess of 0.68, which indicates that they are well-chosen classifications [26]. These CAPs closely resemble the student types described by Anderson et al. [1] who clustered MOOC students based upon the ratio of lectures viewed to assignments completed. In this case we used attempts in place of submissions.

Table 2: Course Activity Profile Clusters: size, #lectures viewed, #quiz attempts, and performance.

<table>
<thead>
<tr>
<th>CAP</th>
<th>Lectures viewed (max:54)</th>
<th>Quizzes Attempted (max:7)</th>
<th>% Certificate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viewer</td>
<td>M:49.57, SD:2.95</td>
<td>M:0.55, SD:0.96</td>
<td>0%</td>
</tr>
<tr>
<td>Solver</td>
<td>M:5.30, SD:7.15</td>
<td>M:7.67, SD:0.77</td>
<td>41.10%</td>
</tr>
<tr>
<td>All-rounder</td>
<td>M:45.23, SD:8.3</td>
<td>M:7.58, SD:0.89</td>
<td>79.19%</td>
</tr>
<tr>
<td>Bystander</td>
<td>M:1.87, SD:2.72</td>
<td>M:1.25, SD:1.43</td>
<td>0%</td>
</tr>
</tbody>
</table>

As Table 2 shows, the all-rounders have the highest rate of certificate completion. For the rest of our analysis we will focus on three categories: viewer, solver, and all-rounder.

4.2 RQ2: CAPs by Country

Does the proportion of student categories differ by country?

After identifying the meaningful CAP clusters, we compared countries based upon the proportion of CAPs observed. We again applied hierarchical clustering on countries with more than 15 users from the viewer, solver, and all-rounder students. In this case we found that three clusters yielded the highest ASW values. These clusters are shown in Figure 2.

This clustering grouped countries with a high proportion of solvers in Cluster 1. This includes developing countries, Russia, and Singapore. The proportion of solvers present in Cluster 1 is significantly higher than that of cluster 3.
4.3 RQ3. Quiz Activity Profiles, QAPs

When do students in each category access the different types of course activities and how is that correlated with quiz submissions?

After identifying the CAPs and examining their relative proportion within countries, we proceeded to analyze the inter-country behavioral differences within each CAP. It is our hypothesis that students from different countries will behave differently given the different Hofstede/CDLF dimensions. In order to assess this hypothesis we analyzed the behavioral differences among users with regards to the course content accessed in the three learning phases described below.

In order to better understand when students engaged in different learning activities we segmented the activity sequences into three phases based upon the quiz attempts. These phases are shown in Figure 3. For each phase we counted average number of lectures viewed (VL), forum posts made (MP), and posts read (RP). For the first quiz submission, and for the subsequent submission phases, we also counted the average number of times that a student attempted and submitted the same quiz (AQ, SQ). We excluded viewers from this analysis as they made little to no attempts at the quizzes.

The relative QAP values for solvers and all-rounders in this analysis are shown in Figure 4. We then conducted a series of pairwise Kruskal-Wallis tests [20] with Benjamini-Hochberg correction [3] comparing the performance by group and learning phase to a baseline of the course average. We found that the solvers and all-rounders viewed significantly more lectures between the quizzes and read more posts during subsequent quiz submissions than in the other learning phases.

4.4 RQ4. QAPs by Culture

How do quiz-based activity profiles and countries relate to the four overlapping Hofstede/CDLF cultural dimensions of: power distance, individualism, masculinity, and uncertainty avoidance?

We applied hierarchical clustering on countries with more than 15 all-rounders, solvers or viewers, based on the four Hofstede dimensions [13] incorporated in CDLF [24]. We found three clusters with ASWs over 0.46. We treated the first cluster as the baseline for comparison since it contains the majority population in BDE. We conducted a Kruskal-Wallis test on the QAPs between clusters, for each course activity in each learning phase. The results are shown in Figure 5.

Countries in cultural cluster 1 (Australia, Canada, the U.S. and U.K. cluster) have the lowest average power distance and the highest average individualism. In our analysis we found that solvers in clusters 2 (Russia, Spain, Brazil, & France) and 3 (China, India, & Singapore) read and made fewer posts during multiple learning phases. These differences were significant or marginally-significant. Moreover, solvers in cluster 3, whose countries are characterized by the highest average power distance and lowest average individualism, viewed significantly fewer lectures between the quizzes. All-rounders in cluster 3 also viewed significantly fewer lectures during the first quiz submission and made more submissions per quiz, this difference was marginally significant.
We found a high degree of overlap between the cultural clusters and the CAP clusters described in section 4.2. Cultural cluster 1 is a subset of the all-rounder CAP cluster, by country, and cultural cluster 3 is a subset of the solver CAP cluster. These results suggest that students from countries with higher individualism and lower power distance are more prone to focus on evaluations. However, cultural cluster 2 includes students that were evenly split between solvers and all-rounders. These findings suggest that the cultural dimensions are directly connected to some aspects of the students’ observed behaviors, but other personal motivations may also dominate student behaviors.

4.5 RQ5. Forum “Best Friend”
Is a student’s most frequent forum partner in the same country/culture?

For this analysis we identified each students’ “best friend” based upon their forum interactions. In a prior study, we tested whether we can predict students’ performance in the course based upon their implicit social relationships in the forum [11]. In this case we constructed a similar relationship graph for the 750 forum users based upon that work and the work of Fire et al. [6]. Edges in the graph were weighted based upon the number of times that a user had replied to a thread that the other used had posted in. We then defined a students’ “best friend” as the individual with the highest-weighted edge between them.

Then, for each of the top 15 countries and the 3 cultural clusters defined in the prior section we performed a $\chi^2$ test with the proportion of “best friends” within the cluster as the dependent variable. Our goal was to test whether or not the cluster was a significant predictor of the proportion of individuals with “best friends” in their cluster. The results are shown in Table 3. We found that for all three cultural clusters, the students are significantly more likely to have a best friend within their own country.

5. DISCUSSION
In this study, we conducted an exploratory analysis on three dimensions of MOOC behavior by country and culture. We first identified five Course Activity Profiles (CAPs) based on the number of lecture views and quiz attempts: viewers, solvers, all-rounders, samplers, and bystanders. We found that the all-rounder students were most likely to obtain a certificate of completion, followed by the solvers. This indicates that the behavior profiles exhibited by these groups are a good indicator of students who are working toward certification.

We then studied the distribution of CAPs over countries. To that end we clustered countries with 15 or more students in the solver, viewer, or all-rounder categories based upon their CAP distributions. Interestingly we found that the developing countries in our dataset all contained a substantially higher proportion of solvers than other countries. We then clustered the same set of countries using the Hofstede/CDLF cultural frameworks [13, 24]. We found that the resulting cultural clusters also aligned with the observed student types. We then clustered the same set of countries using the Hofstede/CDLF cultural frameworks [13, 24]. We found that the resulting cultural clusters also aligned with the observed student types. We then clustered the same set of countries using the Hofstede/CDLF cultural frameworks [13, 24]. We found that the resulting cultural clusters also aligned with the observed student types.
the case that the solvers are studying offline and are using the MOOC as a certification system.

Following that we focused on the students’ quiz-centric behavior. We defined the Quiz Activity Profiles (QAPs) based upon the students’ major activities between quizzes and before subsequent quiz attempts. We found that, regardless of the student’s CAP, they typically viewed lectures between quizzes, and then turned to forum posts after their initial submission and before any resubmission. This resembles some traditional classroom settings where students attend lectures before doing homework and then only turn to the office hours or peers after they face some difficulty.

When clustering the countries by cultural dimensions we also found that two of our clusters were dominated by countries with higher power distances and lower individualism (cluster 2: Russia, Spain, Brazil, and France; cluster 3: China, India, & Singapore). Students in these clusters were less likely to interact on the forum in most of the learning phases than the students in cluster 1 which was dominated by countries with low power distance and high individualism. This finding is consistent with other work on the CDLF which found that students in countries with high power distance tend to treat the teacher as the unchallenged communicator versus students in countries with low power distance who place a higher value on dialogue and discussion in the learning process. This framework, however, does not explain the other observed variations in cultural cluster 3, notably their apparent focus on work between quiz attempts. We believe that the explanation may lie in the educational culture of this cluster. As noted above this cluster consists entirely of Asian nations which are historically test-driven. We believe that this educational culture may cause the students to view quizzes as the primary goal, leading them to focus their efforts on viewing lectures and forums after they have seen the quiz. Moreover, this cultural emphasis on exams may be the primary reason that Asian students were more prone to re-submit quizzes rather than moving on to new material.

Finally, we analyzed students’ “best friend” on the forums. We found that students are more likely to have a “best friend” [6, 11] from countries in the same cultural cluster as their own. Chinese and Brazilian students, in particular, are more likely to have “best friend” from their own country. This close connection may be explained by several factors. First, students from the same country may have the same motivations and overall view of the course which would lead them to join forums that fit their shared needs. Second, students may face difficulties in communicating with individuals from other nations due to language barriers, thus making them more connected to their neighbors. And third, the observed relationships in the forums may reflect real offline relationships among students who joined the class together and are collaborating offline. In the absence of additional data we cannot distinguish among these alternatives.

Ultimately we conclude that students from different countries and cultures do exhibit different learner behaviors on the BDE MOOC. These differences may be explained by country, cultural dimensions, and educational differences. We believe that the students’ observed behaviors are driven in part by their own goals and their unique cultural background. Students who come from countries that value discussion are more prone to interact on the forums. Students who come from countries that are test-centric are more prone to focus on improving their quiz scores and will structure their efforts around that. These findings contribute to our understanding of the role that culture and country play in MOOC learner behaviors. They also suggest some culturally-influenced behaviors that MOOC designers should consider when designing their materials.

5.1 Conclusions & Future Work
Our goal in this study was to increase general understanding of behavioral differences in MOOC populations, and the possible role that country and culture may play. We found interpretable inter-country and intercultural differences in students’ observed activities, both across the whole course and when segmented by quizzes. We also found that forum users were most strongly connected to individuals from their own country or from culturally-related countries. We analyzed these findings in the context of a hybrid Hofstede/CDLF cultural framework and found that our observed clusters were consistent with the theoretical literature.

This paper is one of the first to explore the relationship between observed behaviors and learners’ country or culture. In future work we plan to examine the generality of these findings by analyzing other related MOOCs. Our present dataset includes 29,149 accounts identified from the clickstream data, only 1,591 of which were non-bystanders, and only 750 of whom participated in the forum. While this is consistent with other MOOCs, it is also somewhat skewed and contains relatively small samples for many countries.

As we build a better understanding of the interactions between culture, behavior, and MOOC performance, new questions arise for MOOC designers. Should learning platform designers intervene to change cultural behaviors? For example, should they encourage students to use forums more or to communicate across cultural lines? Or should they support many separate groups by providing language specific forums and tailored tracks? If so, how can we assess the impact of such interventions? It may be worthwhile to conduct more user-centered research so that we can better understand the unique needs of diverse populations. This type of work may help us to better understand how to address the diverse needs of such unprecedented student populations.

6. ACKNOWLEDGEMENTS
The author would like to thank Abhishek Agrawal and Dhvey Shah for segmenting clickstreams and identifying students’ geographical locations in this MOOC. This work was partially supported by NSF grant no. 1418269.

7. REFERENCES


Effect of student ability and question difficulty on duration

Yijun Ma
Teachers College, Columbia University
New York, NY 10027
ym2476@tc.columbia.edu

Lalitha Agnihotri
McGraw Hill Education
2 Penn Plaza
New York, NY 10121
lalitha.agnihotri@mheducation.com

Ryan Baker
Teachers College, Columbia University
New York, NY 10027
ryanshaunbaker@gmail.com

Shirin Mojarad
McGraw Hill Education
281 Summer Street
Boston, MA 02210
shirin.mojarad@mheducation.com

ABSTRACT
Time has become a standard feature used in EDM models, and is used in models of meta-cognitive strategies to models of disengagement. Most of these models consider whether a student action is “too fast” or “too slow”. However, an open question remains on how we define and select these cut-offs. Moreover, it is not clear that the same cut-offs are appropriate across different situations. Some students may generally respond faster than others; more difficult items may take different amounts of time. In this paper, we consider whether absolute or relative indicators of time are more appropriate as cut-offs, and whether simple transformations (such as log time) are useful when representing time. We do so through visualizing student performance in relation to general student ability, item difficulty, and different ways of representing time. We find that student knowledge and item difficulty should be taken into account when choosing cut-offs, and that there are advantages to representing duration in terms of standardized log-time.

Keywords
Time taken, Duration, Visualization, Student Ability, Rasch Model, IPI, Item Difficulty

1. INTRODUCTION
Over the decade since the Educational Data Mining community began to coalesce, one of the most common ways to interpret student behavior has been to look at the amount of time taken to respond to questions. Early work by Aleven, Baker, and Beck tried to determine whether a response was “too fast”, indicating gaming the system, help abuse, try-step abuse, or disengaged behavior [1, 2, 3]. Soon, work began to consider whether a response was “too slow” as well [4]. Researchers noted that performance seemed to degrade when behavior reached either of these two extremes. This theme of trying to identify behavior as “too fast” or “too slow” continues to this day [5, 6]. Actions that are “too fast” or “too slow” are seen as components in a range of EDM models, including contemporary models of gaming the system [7], off-task behavior [8, 9], carelessness [10, 11], and self-explanation [12]. However, one of the interesting aspects of this body of literature is how remarkably inconsistent it is, as noted by [5]. Despite their conceptual simplicity, researchers do not agree what “too fast” or “too slow” means. This inconsistency may not be a major concern when these parameters are empirically fit using training labels, but is somewhat more concerning when cut-offs are rationally defined.

Part of the reason for inconsistency, of course, is that “too fast” and “too slow” are inherently contextual. Interfaces matter. A student completing division problems by typing in answers is likely to respond faster than a student chancing down a skeleton and hitting the right divisor key [13]. Ability matters. A 7-year old solving arithmetic problems is likely to perform more slowly than a 38-year old. Difficulty matters. Even for the same user interface and an experienced adult, “49 / 7” will be solved more quickly than “602 / 7”.

For this reason, it is unlikely there is a universal answer to how fast is too fast, and how slow is too slow. Nor will it be easy to find a simple formula or set of formulas that can predict this. Mathematical models based on memory [14] can make predictions about speed in some situations, but are incomplete for many of the complex types of problem-solving and the activities surrounding problem-solving in modern learning environments. At the same time, there exist simple psychometric models that can predict a considerable amount of variance in performance, which may be useful in investigations of this nature.

One solution, as discussed above, is to empirically select a single cut-off, but part of the challenge is that even within a learning environment, cut-offs both vary contextually, and exist on a continuum. In this paper, we will examine this continuum in a visual fashion, across different situations within a single online learning environment. Specifically, we will analyze how the relationship between time and performance varies when students vary in knowledge, and for items of different overall difficulty.

We will also investigate whether the most commonly used way to represent time (number of seconds) is the best representation for understanding these issues, or whether standardizing or transforming time makes it easier to understand the relationship between time and performance.
By better understanding these relationships, we will be able to select more appropriate cut-offs, and develop more precise models for discovery with analysis and interventions.

2. DATA SET

We investigate these issues in the context of one of the world’s most widely used digital learning environments, McGraw-Hill Education’s Connect system [16, 17]. Connect is currently actively used by approximately two million students and 25,000 instructors. Within Connect, instructors select questions from question banks and the system then administers them to the student as homework, quiz, exam, or practice assignments. Most items are auto-graded by the system, and immediate feedback is provided when relevant (e.g. not during exams). Within homework and practice assignments, students can make multiple attempts to answer each question, based on the policies set up by the instructor. In this paper, we use item and questions interchangeably.

Connect is organized into courses; each course is tied to a McGraw-Hill book title, and question banks are organized in relation to book chapters. In this paper, we focus on a single textbook in order to avoid including radically different material together in the same analysis (for example, one might expect calculus problems to take longer to solve than questions about the factual aspects of history). We analyze a data set from 173 courses that utilize the title McGraw-Hill’s Taxation of Individuals and Business Entities, 6th Edition, by Brian Spilker, a medium-sized data set with relatively consistent item design, involving a course text with items selected as a focus for enhancement within McGraw-Hill at the time this research was being conducted. Within this textbook, there were multiple types of items: multiple choice items where single responses were correct, multiple choice items where multiple responses were correct, fill-in-the-blank items, matching questions, and ungraded essays (removed prior to analysis).

Within this textbook, within the period between August 2014 and November 2014, 3,882 students (working with 86 instructors) answered 2,947 distinct questions. In total, this set of students attempted to answer questions 536,520 times, an average of 138.21 attempts per student.

Prior to analysis, we removed all ungraded questions from the data set (as assessing correctness is outside the scope of this paper). We also removed attempts where the student timed-out due to inactivity within the system for 60 minutes, and where the student’s response time was not collected or had impossible values (due to logging errors). For this specific analysis, we removed students’ second and subsequent attempts to answer questions, focusing on their performance and time taken on their first attempt. Although second and subsequent attempts are relevant to issues of modeling student behaviors such as off-task behavior and gaming the system, these times are strongly influenced by the time taken on the first attempt, and are relatively more complex to consider. As such, we leave analysis of second and subsequent attempts to future work. The resultant cleaned data set involved 3,632 students answering 2,689 distinct questions, attempting to answer items 365,302 times, an average of 100.58 attempts per student.

Within these items, scores were distributed between 0 and 1, with 76% of items receiving a fully correct score of 1. However, partial credit was assigned by instructors and, as a result, is somewhat non-uniform; different items had different partial credit assigned for different responses. As such, the partial credit information was less useful for analysis than in other systems where it is assigned in a consistent fashion [15, 18]. To avoid having our results impacted by this inconsistency, we assigned a value of 0 (incorrect) to any student response that was not fully correct. Only 7.9% of the problem attempts were affected by this modification.

2.1 Tagging with Question Difficulty and Student Ability

In order to understand how student knowledge and item difficulty influence the relationship between time taken and performance, we annotated the data with a well-known psychometric model: the Rasch Model [19, 20].

The Rasch Model is one of the most widely used models in the history of psychometrics. It relates performance to student ability (treated here as overall knowledge of the domain) and item difficulty. More recent and advanced models from the psychometrics and student modeling literature consider change in knowledge over time, group items into latent skills, explicitly model the probability of guess and slip, and use different uncertainty functions for students and items [21, 22, 23, 24, 25]. However, the Rasch model is appropriate for the analysis here, as assesses student knowledge and item difficulty (which is what we focus on in the analyses below), it is known to function well when different students answer different items [19], and has high stability and reliability [20].

The equation for the Rasch model is given as follows [19]:

\[ P(\theta) = \frac{1}{1 + e^{-1(\theta-b)}} \]

where \( b \) is the question difficulty parameter, \( \theta \) is the student ability (knowledge) level, and \( P(\theta) \) is the probability that the student will answer the current item correctly. Within this model, if a student’s ability is equal to the item’s difficulty (\( \theta = b \)), the probability that the student will answer the question correctly is 50%. As the student’s ability becomes higher or the item’s difficulty becomes lower, the probability of correctness increases and finally is approximately equal to 1; correspondingly, as ability becomes lower or difficulty becomes higher, the probability of correctness approaches 0.

As is standard [19], we use Maximum Likelihood Estimation, in this case converging after seven iterations, to estimate the values of \( \theta \) and \( b \) for each student and item based on actual data. After fitting and applying the model, all student attempts are tagged with a difficulty parameter and an ability parameter.

This model achieves an R-squared value of 0.322, and an A’ (mathematically equivalent to AUC but easier to calculate) of 0.852, calculated using the A’ calculator available at http://www.columbia.edu/~rsb2162/computeAPrime.zip.

3. Analysis

We analyze the research questions discussed above through a set of visualizations, created in Python’s matplotlib library. Each of the visualizations will place some variant of the time taken by the student to give a response on the X axis, and place the percentage of times when the student response was correct (percent correct) on the Y axis. In the visualizations, item responses are binned to one-second grain-size. For that
bin, we find the percent correct and plot a dot there; if there are more items in the bin, the dot is made larger.

### 3.1 Baseline Graph

In the first visualization, Figure 1, we consider the baseline relationship between time taken and percent correct. Item difficulty according to the Rasch model is also included in the visualization as color, with darker colored dots representing easier items and lighter dots representing harder items (e.g. if a dot is dark, the items composing that dot were on average easier).[12]

![Figure 1: The relationship between the time taken to respond to an item, and correctness. Color is used to denote item difficulty.](image)

As Figure 1 shows, students who spend very little time on an item typically achieve low percentage correct. As the time taken increases, performance improves, curving up from 0 seconds to about 12 seconds; this range of the graph is denoted “A”. Percent correct remains stable from 20 seconds to 60 seconds; this range of the graph is denoted “B”. As students spend over 60 seconds, their performance somewhat declines again; this range of the graph is denoted “C”. This graph shows a similar qualitative pattern to the pattern seen in other systems, but with the shifts occurring at different points. For example, Beck [3] finds that performance improves up until the student has spent 4 seconds, remains stable under 7 seconds, and drops gradually after that.

It is worth noting that despite these shifts, it is non-trivial to find cut-offs. 12 seconds is approximately the inflection point where performance shifts to being stable, but it probably contains more positive behavior than would be desired. It might still be desirable to pick a lower cut-off point for “too fast”. Similarly, the difference between 60 seconds and 100 seconds for “too long” is relatively minimal.

One limitation to Figure 1 is that fewer and fewer data points are seen as the times get longer, making it difficult to show all the data in a relatively limited horizontal space. This limitation can be addressed by switching from absolute time in seconds, to a logarithmic scale for time, shown in Figure 2. By switching to a logarithmic scale, the long tail of long response times is compressed to a small section of the plot and we can show more data while maintaining the essence of the graph. The log scale thus makes it easier to present our full data.

The log scale also makes it easier to see that there are more inflection points than Figure 1 showed. The same ranges (0-12 seconds, 20-60 seconds and 60+ seconds) are marked in Figure 2 as in Figure 1, to enable comparison. Note that between 0-12 seconds (range A), there is a secondary inflection point around 3.5 log time taken where performance shifts from improving slowly to improving quickly. This might be a better cut-off for “too fast” than 12 seconds. Similarly, the decline in performance can be seen to begin around 4.75 log time taken but to accelerate after 5.5 log time taken, suggesting a potentially better “too slow” cut-off. While these cut-offs are somewhat harder for a reader to interpret directly from the numbers, they allow us to make more sophisticated distinctions than were possible just from absolute time.

![Figure 2: The relationship between the time taken (log scale) to respond to an item, and correctness. Color is used to denote item difficulty.](image)

### 3.2 Standardization

One common decision seen in many models that measure student time [26, 27] is to represent student time in terms of standard deviations faster or slower than the average time, calculated as a Z-score, and referred to as standardized time or unitized time. This transformation, which assumes that time is normally distributed, uses the formula

\[ Z = \frac{\text{Time} - \text{Mean(Time)}}{\text{SD(Time)}} \]

The logic is that this approach accounts for the fact that different items need different amounts of time to answer them, allowing fairer comparison of student time on different items.

Figure 3 shows the results of applying this transformation to our data.

![Figure 3: The relationship between the standardized time taken to respond to an item, and correctness.](image)

As this graph shows, most of the data is now clumped together. Notably, the center of the data is not at 0 SD; instead the median is somewhere around -0.5 SD. Though 0 SD is by definition the average value, it is clearly not the median value. This is a common limitation to using standardization, and one that the authors have observed in previous data sets as well. As such, using standardization is vulnerable to skewness and outliers in the original data, making it broadly unsuitable for use across data sets – or indeed, for cases where the magnitude of the long time outliers may vary over time. This can occur, for example, when the original data set has a small number of students with extremely high outlier times, or when the system time-out may change over time. This suggests that standardized time is undesirable for use in cut-offs, since the cut-off points may vary depending on the exact outliers in the data set. This could be addressed by ignoring the outliers when computing...
the SD value (i.e. truncating the values of extreme outliers [28];) but doing so will only incompletely address a second problem; the data is highly compressed relative to the previous visualizations we have examined. Most of the data points occur in a fairly small range. In this case, 64.4% of the data is clumped between $Z = -1$ and $Z = 0$. If the data were distributed according to assumptions, 68% of data would be clumped between $Z = -1$ and $Z = 1$, double the range. This clumping makes it difficult to see the inflections in performance for rapid student responses; although the graph’s clumping does allow us to see that there is some rise in performance for very high response times (a set of outliers outside of bounds for the earlier representations).

One alternative, shown in Figure 4, is to use [29] modified Z-score, which is computed as:

$$M_i = \frac{0.6745 \text{ (Time} - \text{Median(Time))}}{MAD \text{ (Time)}}$$

where MAD stands for Median Absolute Deviation.

This approach centers the data better, but does not solve the problem of the data being compressed.

Another alternative is to conduct standardization on time transformed to a logarithmic scale, shown in Figure 5. As we saw in the previous section, using a logarithmic scale spread out the data better and allowed us to see inflection points more clearly.

### 3.3 Studying Item Difficulty

One factor that is worth considering is that the time taken appears to be associated with how difficult the items are. Figures 1 and 2 each show difficulty in terms of color, with blue representing easier items (according to the Rasch model discussed above) and white representing harder items.

In Figure 1, we can see that the hardest items are found at the two ends of the spectrum; the briefest times taken, and the longest times taken. It is unsurprising that students take longer on hard items. The connection between difficulty and brief responses is also reasonable; students are more likely to become disengaged and engage in behaviors such as gaming the system and carelessness when encountering hard items [30]. The same pattern is seen in Figure 2, although whether the lowest difficulty is seen for higher or lower times varies between graphs. This is simply a result of the fact that Figure 2 shows more of the data set than Figure 1, due to the use of a logarithmic scale.

This leads to the question of how we should expect the relationship between the student’s time taken and their performance to change based on item difficulty. In particular, does the same amount of time taken mean different things for easy items versus difficult items? It is plausible to hypothesize – for example – that rapid responses on easy items may imply fluent knowledge [31] but rapid responses on difficult items may imply disengagement [3].

We examine this by grouping items, based on their difficulty according to the Rasch model $b$ parameters, into 5 bands, shown in Table 1, and displayed in Figures 6 and 7.

<table>
<thead>
<tr>
<th>Difficulty Groups</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficulty $&lt;-1$</td>
<td>Dark Blue</td>
</tr>
<tr>
<td>Difficulty 0 to -1</td>
<td>Light Blue</td>
</tr>
<tr>
<td>Difficulty 0 to 1</td>
<td>Light Yellow</td>
</tr>
<tr>
<td>Difficulty 1 to 3</td>
<td>Yellow</td>
</tr>
<tr>
<td>Difficulty $&gt;3$</td>
<td>Red</td>
</tr>
</tbody>
</table>

As Figure 5 shows, standardizing using a logarithmic scale centers the data as well as using modified Z-score, but spreads the data out better. The data is broadly centered on $Z = 0$, with most of the data (68.82%) between $Z = -1$ and $Z = 1$ (almost exactly the amount that one would expect for normally distributed data). The same inflection points visible at the left side of Figure 2 are visible at the left side of Figure 5. At the same time, while the logarithmic nature of the transformation does compress the right tail somewhat, we nonetheless can see the same rise in performance at very high time taken that we saw in Figure 3. As such, this representation helps us in understanding the data and choosing cut-offs, while gaining the benefit of comparability that standardizing variables gives us.

As Figure 6 shows, the pattern for dark blue and light blue (the lower-difficulty items) is largely the same as in Figure 2. Correctness increases fairly rapidly when students spend more time, leveling off and then slowly declining for high amounts of time spent. However, the amount of time needed for high
levels of correctness is higher for the light blue items ($b$ between 0 and -1) than for the dark blue items ($b$ below -1). This suggests that the same cut-off for “too fast” is not appropriate for items with different difficulty.

As Figure 7 indicates, this difference between the time needed for the lowest-difficulty items (dark blue) and the moderately low-difficulty items (light blue) cannot be controlled for, simply by switching to standardized log time. Even after we switch to standardized log time, more time is needed for the moderately low-difficulty items than for the lowest-difficulty items, to reach high levels of correctness.

The decline in performance for students who spend too much time (possibly going off-task, or asking for help) is seen for both of these two item difficulty groups, in both the log-time graph and the standardized log-time graph.

Interestingly, the patterns seen are different for the higher-difficulty items. Focusing on yellow and red, we can see that there is no clear inflection point where spending more time is associated with worse performance, or even a clear leveling off in performance. For yellow ($b$ between 1 and 3), there is a range between -1 and -1.5 standardized log time where performance may be leveling off or mildly dropping, but it is at best a minor and brief shift, compared to the lower-difficulty bands. For yellow, “too fast” cut-offs could be placed within the -1 to -1.5 SD range, somewhat higher than for lower difficulty (it is hard to identify any good place for a cut-off in the non-standardized graph). For red ($b$ above 3), there is essentially no range where increasing time does not improve performance. For neither of these bands is there a clear “too slow” range, where performance worsens once too high a time spent is reached.

These graphs show that time cut-offs should not be considered independently of item difficulty. We are not aware of any models of gaming the system, carelessness, off-task behavior, or related constructs that explicitly consider item difficulty. Our results suggest that this omission is lowering the quality of these models.

3.4 Studying Student Knowledge

Finally, we consider how the student’s knowledge of the domain impacts their time spent. Figures 8 and 9 each show knowledge in terms of color, with green representing more knowledgeable students (according to the Rasch model discussed above) and white representing less knowledgeable students. Note that this color scheme corresponds to the color scheme used for difficulty – students are less likely to produce correct answers for white dots.

This result suggests that less knowledgeable students appear to be more likely to engage in behaviors such as gaming the system and carelessness, but there does not seem to be a similar pattern for off-task behavior.

Figures 8 and 9 show a different pattern than Figures 1 and 2. Whereas those earlier figures indicated that short and long times were seen for hard items, Figures 8 and 9 indicate that brief times are seen for the least able students while long times are generally seen for knowledgeable students. This result suggests that less knowledgeable students are generally less likely to engage in behaviors such as gaming the system and carelessness, but there does not seem to be a similar pattern for off-task behavior.

Figure 10 shows the same item difficulty bands as were seen in Figure 7, but colored in terms of student ability rather than item difficulty. We can see that regardless of question difficulty, if the response time is too fast relative to the average for the item, the student is likely to be of low ability. However, we can also see from box T1 that this low ability is also seen for longer response times for harder items. For the easiest items, lower ability is seen below -2 SD for time; for the hardest items, lower ability is seen below -1.2 SD for time. As such, this figure indicates that the behavior of answering too fast is seen across questions with different difficulties, though the cut-off should differ.

For higher difficulty items, longer time taken is associated with better students, as shown in T2. But this effect only manifests for the higher difficulty items; these items are more discriminative in terms of the relationship between student ability and longer time taken. Finally, most of the examples of responses that are relatively much longer than other responses occur on the easier items – it is harder to distinguish responses that are genuinely too long for harder items.
Figure 10: The relationship between the log-transformed time taken to respond to an item, and correctness, for each of the difficulty bands shown in Table 1, but colorized in terms of student ability.

Given these results, we can reasonably ask: how should we expect the relationship between the student’s time taken and their performance to change based on the student’s general knowledge of the item? In particular, does the same amount of time taken mean different things for knowledgeable students versus not knowledgeable students? Correspondingly, with the above, it is plausible to hypothesize – for example – that rapid responses by knowledgeable students may imply fluent knowledge but rapid responses by struggling students may imply disengagement [14].

We examine this by grouping students, based on their knowledge level according to the Rasch model $\theta$ parameters, into 5 bands, shown in Table 2, and displayed in Figure 11.

Table 2: The difficulty groups shown in Figure 11, based on $b$ in the Rasch model. Items with $\theta$ below -3 look very similar to items with $\theta$ from -1 to -3, so they are included in the same group.

<table>
<thead>
<tr>
<th>Knowledge</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt;-1$</td>
<td>Dark Red</td>
</tr>
<tr>
<td>0 to -1</td>
<td>Brick Red</td>
</tr>
<tr>
<td>0 to 1</td>
<td>Pink</td>
</tr>
<tr>
<td>1 to 3</td>
<td>Light Green</td>
</tr>
<tr>
<td>$&gt;3$</td>
<td>Green</td>
</tr>
</tbody>
</table>

Figure 11: The relationship between the standardized log-transformed time taken to respond to an item, and correctness, for each of the student knowledge bands shown in Table 2.

As Figure 11 shows, the pattern for brick red, pink, and light green (the medium-knowledge students) is largely the same as in Figure 9. Correctness increases fairly rapidly when students spend more time, leveling off, declining, and then coming back up a little for the highest amounts of time spent. The pattern is different for the highest-knowledge students.

The highest-knowledge students (green) essentially do not have any very rapid responses and show similarly high performance across the spectrum of time taken. This can be interpreted in at least three ways. Perhaps the highest-knowledge students do not become disengaged; alternatively, perhaps the students who never become disengaged perform better, and appear to have the highest knowledge. Or perhaps being classified by the Rasch model as having the highest knowledge requires both having the highest knowledge and never becoming disengaged.

The lowest-knowledge students (dark red) have very poor performance for low amounts of time spent. However, their performance never flattens out, although the rate of improvement slows. The more time these students spend, the better they do. Despite that, these students’ performance never reaches a very high level.

One other thing that is visible in the graph is that the amount of time needed for asymptotic levels of correctness is lower for the higher knowledge students ($\theta > 1$) than for the lower knowledge students ($\theta < 0$). See the line B-D in the Figure, which links the asymptotic point for high-knowledge students to the near-asymptotic point for low-knowledge students. This suggests that the same cut-off for “too fast” is not appropriate for students with different ability.

4. DISCUSSION AND CONCLUSIONS

In this paper, we have investigated how the relationship between the time taken by students and their performance is mediated by student general knowledge and item difficulty. We also investigate whether different ways of representing time (standardized or non-standardized; log-transformed or non-transformed) impact our ability to recognize cut-offs and inflections in student performance. We analyze these questions by visualizing the relationship between time taken and performance under each of these different conditions.

We find that using a logarithmic scale allows for showing more data while making it easy to present the full data range while standardization allows for a fairer comparison of student time on different items. We find that the combination of these approaches facilitates identifying cut-offs and infection points in student performance.

We find that students who spend very little time on an item typically achieve low percent correct and as the time taken increases, performance improves. However, as students spend over a certain time, their performance somewhat declines again. The amount of time needed for very successful performance is different for easier and harder items and is higher for the easy items compared to very easy items. Hence, we suggest that the same cut-off for “too fast” is not appropriate for items with different difficulty levels.

Student performance declines when students spend too much time on easy and very easy items. The patterns seen are different for the higher-difficulty items. For the difficult and very difficult items, we do not observe any clear inflection point where spending more time is associated with worse performance.

As such, we can conclude that time cut-offs should not be considered independently of item difficulty. We are not aware of any models of gaming the system, carelessness, off-task behavior, or related constructs that explicitly consider item difficulty. Our results suggest that this omission is lowering the quality of these models.

In terms of student overall domain knowledge, we find that the most successful students seldom respond in very short amounts of time. As discussed above, this may reflect in part the fact
that very quick responses make the student appear generally less successful within the Rasch model. However, we also see that the generally knowledgeable students show consistently high performance for most the span of time taken, whereas the less generally knowledgeable students' performance does not level off to the same degree.

For higher difficulty items, longer time taken is associated with better students. However, this effect only manifests for the higher difficulty items; these items are more discriminative in terms of the relationship between student ability and longer times taken. In future work, we will try to correlate these longer times with students' usage of other online materials during. At present we do not have access to this level of detailed data.

These results suggest overall that models that consider student time taken during online learning, and select time cutoffs, should take student general knowledge and item difficulty into account. However, the exact cut-offs will probably differ between systems and also possibly differ with content.

It would be useful to investigate whether the findings seen here are general across other contexts. In our future work, we will investigate their generality to other textbooks, and whether the findings also generalize to other online learning platforms. It would also be useful to examine existing models depending on time cutoffs, and see whether measures of general student knowledge (perhaps average correctness so far across skills) and item difficulty can produce more accurate models of constructs like gaming the system and off-task behavior. Ultimately, this type of model may enhance the effectiveness of behavior detection, leading to more effective interventions to struggling and disengaged students. One of our upcoming steps will be to use these analyses to develop behavior detectors for our platform, that can be used to help to students who are answering too fast or who are struggling and responding slowly. We will then measure the impact of these changes on learning outcomes, to see the degree to which these approaches can enhance student learning.

5. ACKNOWLEDGMENTS
Our most sincere thanks to Mark Riedesel and Alfred Essa for supporting this research. We would also like to thank Malcolm Duncan and his team at the EZTest who helped us navigate through the database and helped create queries to pull the data required for this research.

6. REFERENCES


Modeling the Influence of Format and Depth during Effortful Retrieval Practice

Jaclyn K. Maass
University of Memphis
Institute for Intelligent Systems
Memphis, TN 38152
1-352-584-9103
jkmaass@memphis.edu

Philip I. Pavlik Jr.
University of Memphis
Institute for Intelligent Systems
Memphis, TN 38152
1-901-678-2326
ppavlik@memphis.edu

ABSTRACT
This research combines work in memory, retrieval practice, and depth of processing research. This work aims to identify how the format and depth of a retrieval practice item can be manipulated to increase the effort required to successfully recall or formulate an answer, with the hypothesis that if the effort required to answer an item is increased there will be more benefit to learning. This hypothesis stems from work on desirable difficulties and the effortful retrieval hypothesis. Our data source was an experiment that used a 2 (question depth: factual, applied) x 2 (answer format: multiple choice, short answer) between-subjects design to investigate the effects of these conditions on retrieval practice performance. The experiment was delivered online though Mechanical Turk (n = 178). A logistic regression predicting performance during practice indicates that participants get more (in terms of an increase in future predicted success) from successful retrievals of items that fall within the more difficult level of both the format and depth factors (i.e., short answer and applied). There is also some support that the benefit from multiple choice items may be increased by asking deeper, more applied questions. The application of these results to scheduling effective practice is discussed.

Keywords
Retrieval practice, application, difficulty, multiple choice, short answer, modeling, depth of processing

1. INTRODUCTION
The testing effect is the well-replicated benefit of retrieval practice (i.e., “testing yourself”), typically over the course of several repetitions [e.g., 1; 7; 16; 30; 33]. Experiments often compare the benefit of active retrieval against re-reading or re-studying written material and much of the early work in this field utilized a more traditional cognitive psychology experimental setup (e.g., using word lists/pairs or isolated facts, controlling for prior knowledge, and post testing with verbatim items repeated from practice). This design, however, does not well represent how retrieval practice would be implemented in authentic educational settings. For implementation in classrooms, issues that have real-world importance to educators, such as the format of the questions and the ease of administration, should be considered.

The effect of answer format has long been of interest not only to educational researchers (e.g., comparing multiple choice, fill-in-the-blank, essays, etc.), but also to cognitive psychologists (e.g., comparing recognition, cued or free recall, etc.). Research has shown a continuum in terms of performance/difficulty ranging from recognition, to cued recall, to free recall which translates roughly in educational terms to multiple choice, short answer, and essay questions. This ordering is found consistently in research and is summed up nicely by Glover’s [13] work which reported the effectiveness of three formats used during retrieval practice (referred to as intervening tests): free recall, cued recall, and recognition (see Experiment 4). After reading a passage and having intervening tests in one of the three formats, participants took a retention test after four days. The free recall intervening test was an open-ended format, with participants writing what they remembered from the passage. The cued recall intervening test was a fill-in-the-blank format, using sentences paraphrased from the text. The recognition intervening tests required the participants to identify which of several sentences they had read previously in the text. The final retention test included items in each of the three formats (across the posttests in Experiments 4a, 4b, and 4c). A very clear pattern emerged: the fewer cues there were available during practice (e.g. free recall provided the fewest cues), the better participants performed on the final retention test. Those who had intervening tests in a free recall format out-performed participants in the cued recall condition on the final retention test (statistically significant difference), who in turn outperformed those who practiced with a recognition task (not statistically significant). Perhaps most importantly, this advantage held regardless of the format of the retention test, which included all three formats [13].

There are several other studies which show us the benefit of using fewer cues (e.g., short answer format) during retrieval practice. Kang, McDermott, and Roediger III [18] had participants read several journal articles. After reading each article, participants completed one of four possible tasks- a multiple choice test, a short answer test, reading relevant facts from the text, or a questionnaire (i.e., filler task). When feedback was provided during the practice tests, those items that had been practiced in short answer format had significantly higher scores on the final test. Results also indicated that practice with multiple choice testing was no better than re-reading relevant facts. The researchers concluded with a recommendation for practice testing with short answer items. Similar results were found in work by McDaniel, Anderson, Derbish, and Morrisette [22], which indicated that weekly practice tests were more effective in increasing final test performance when the weekly practice was in the form of short answer questions compared to multiple choice items. Since the final test was only in multiple choice format, it suggests another benefit of short answer
practice is the ability to overcome transfer-appropriate-processing effects, which would predict that the final test performance would be highest when it matched the conditions of earlier practice [24]. In other words, short answer may be a better alternative to multiple choice regardless of how you assess it.

One possible reason for why practice with short answer often outperforms multiple choice on final outcome measures is the amount of effort required for retrieval [18]. This general benefit of effortful retrieval has been referred to as the retrieval effort hypothesis, which was motivated by Bjork’s [4; 5] desirable difficulty framework and Craik and Lockhart’s [11] depth of processing research. The retrieval effort hypothesis, as defined by Pyc and Rawson [29], claims that there is more memorial benefit from successful retrieval practice when it is difficult than when it is less difficult. This follows from the desirable difficulty framework, which suggests that practice which is made more difficult (up to a certain point) will lead to more durable and generalizable learning [4]. The desirable difficulty framework sets a theoretical upper bound on the level of difficulty appropriate for effective learning, which can depend on several individual differences including prior knowledge and working memory capacity. This is similar to the assistance dilemma [19], which suggests there is an optimal middle-ground in terms of how difficult a task should be, and/or how much assistance should be offered to a student during a learning task.

The goal of the current work was to generate data to further investigate the effect of effortful retrieval practice, and specifically, how we can equate the effort required to successfully answer multiple choice items with the effort required for short answer items. One way to address this is to increase the effort required to correctly answer a multiple choice question, and the way to do so may lie within the depth of processing required to respond to the question. By asking a deeper, more applied question, rather than the more common text-based factual question, perhaps we can encourage deeper processing so as to increase the effort required for multiple choice questions.

The depth of processing framework suggests that information which is processed on a deeper level will be encoded in a more elaborate and durable manner, with depth referring to greater semantic or cognitive processing [11]. Craik [10] further defines depth as “the qualitative type of processing carried out on the stimulus…” (p. 307). Questions that require more cognitive processing to successfully answer have also been referred to as deep-processing questions. Deep-processing questions rely on a student’s logic and reasoning abilities and are thought to tap into more complete and coherent understanding [14]. Deep-processing questions are embedded in the deeper levels of cognition in Bloom’s [6] taxonomy, and both have been shown to be positively correlated with final examination scores [14]. In the current work we attempt to increase the difficulty of multiple choice items by asking deeper, more applied questions, and mine our data to compare the benefit that we see from these more difficult multiple choice items with typical benefit from asking factual short answer items.

The interaction of answer format and depth of processing has been investigated to some degree in work by Smith and Karpicke [31], which compared three answer format conditions: multiple choice, short answer, and hybrid conditions which consisted of short answer-multiple choice pairings. Question type during retrieval practice (i.e., factual and inference questions) was a within-subjects factor (Experiments 1, 2, and 3), but this factor was collapsed in the analyses of final assessment performance. They concluded that practice with short answer could lead to higher performance on the final assessment (compared to practice with multiple choice questions), if students achieve a high proportion of correct short answer responses during practice. Smith and Karpicke therefore attempted to equate the initial practice performance between the short answer and multiple choice conditions. Those results are discussed in more detail in their paper [31], but of importance to the current work is that they attempted to raise performance on short answer questions up to the performance on multiple choice items. The current work will essentially attempt the opposite—increasing the difficulty (or lowering the performance) of multiple choice in an attempt to “equate” it to short answer. Therefore, the design of the current data collection was partially inspired by that of Smith and Karpicke, in an attempt to get more fine-grained information about the interaction between format and depth during practice, and their effect on different format and depths at posttest.

In theory, the multiple choice questions in Smith and Karpicke’s work were more difficult when the multiple choice was an inference item, rather than factual, but the nature of their inference questions appears to be fairly straightforward, without requiring much more effort than the factual questions. Specifically, the inference items required participants to combine different facts they had previously read in order to draw a conclusion/answer that had not been explicit in the text. However, for most (if not all) of the inference items, the facts required to answer them were presented within a single paragraph. This is not inherently problematic, but it is important to take note of if your objective is to increase the effort required to answer a multiple choice item, since it brings into question the level of difficulty of the inference questions. For example, an inference would be more difficult to make if it required retrieving and combining more than two facts, or if those facts were presented further apart from each other in the text. Further, the answer options in Smith and Karpicke’s multiple choice items only included a single option that appeared in the text— the correct answer option. Thus, these questions become purely a measure of memory (of a previously read text), rather than understanding or learning. In other words, the students wind up asking themselves, “Which of these answer options did I see in/ matches with the text I read earlier” rather than, “Which of these options make sense and accurately reflects what I read?” This only serves to further reduce the difficulty of multiple choice practice. To alleviate this, the multiple choice answer options for the current work were all feasible, text related answers that underwent several iterations, described in detail in the materials section.

1.1 The Current Study

The current study focuses on two ways to increase the difficulty of retrieval: through the amount of retrieval cues available (i.e., the answer format: multiple choice or short answer) and through the depth of processing required to successfully answer the question itself (i.e., the question depth: factual or applied). We attempt to mine our data to determine whether or not the difficulty of multiple choice be increased by asking a deeper question, and whether difficulty created through varying amounts of retrieval cues (i.e., the answer format) is similar to the difficulty created through the depth of the question.

The purpose of this paper is to investigate the effect of question format, depth, and individual differences during retrieval practice. Although the experiment tested several types of transfer at the posttest (e.g., format, depth, and unpracticed information), this paper is predominantly focused on dissecting the mechanisms at play during practice. In order to do this, we employed a method of model-based discovery [3] in which previously developed models are adapted to fit the particular research questions and data being
mined. In order to create a more complete picture, however, some descriptive information regarding posttest performance is provided, although it is not the main focus of this paper.

2. METHODS

2.1 Design

The experiment manipulated difficulty of retrieval practice through a 2 (question depth: factual, applied) x 2 (answer format: multiple choice, short answer) between-subjects design. The difficulty of the posttest was also manipulated with a 2 (posttest question depth: factual, applied) x 2 (posttest answer format: multiple choice, short answer) x 2 (concepts: practiced, unpracticed) fully factorial within-subjects design. This resulted in four between-subjects retrieval practice conditions (Factual MC, Applied MC, Factual SA, or Applied SA), and posttest questions in each of those four conditions, allowing for measures of transfer to a different depth and format, as well as transfer to previously unpracticed concepts. Prior knowledge was assessed by a 6-item pretest on factual questions, half randomly assigned per participant to multiple choice and half to short answer format. This experiment did not include a control condition with no retrieval practice. This was a conscious decision since the testing effect is widely accepted as a reliable phenomenon, and the current design allows for a more tractable, and fine-grained investigation of specific components of retrieval practice.

2.2 Participants

One hundred ninety-three participants were recruited through the Mechanical Turk (MTurk) online data collection platform. The only requirements were for the participants to be at least 18 years of age, a native English speaker, from the United States or Canada, and be a reliable MTurk worker. The last requirement was defined as a worker who had completed at least 50 MTurk tasks with at least a 95% approval rate. Data for 10 participants were removed due to the participants having ten or more time-outs during the experiment and five participants’ data were removed due to glitches in the system (n=178, 58% male). Within this sample, 45% were in the age range of 26-34 years, 31% were in the age range of 35-54 years, 30% were between 18-25 years, and 4% were between 55-64 years. Most participants reported that their highest level of completed education was “Some college” (37.2%), followed by “High school/GED” (17.7%), “Graduate degree” (6.6%), and “Less than high school” (<1%). Each MTurk worker was paid $5.00 for participation.

2.3 Materials

2.3.1 Text

The experimental text was 995 words in length and pertained to the circulatory system. It was compiled from texts used in previous research [15; 35], and is estimated to be at a Flesch–Kincaid 6th grade reading level (https://readability-score.com).

2.3.2 Factual and Applied Items

Sixteen concepts were extracted from the text to be used for the creation of factual and applied questions. These concepts represent what we believe to be the crucial components in the text, and are aligned with, and expanded from, the factual questions previously used with these materials [23; 35]. The first author, along with another graduate student familiar with this line of research, created a factual and an applied question based on each of the 16 key concepts. The factual versions for the 16 concepts are taken directly from the text. For example, the text states, “The heart is a pump. Its walls are made of thick muscle. They can squeeze (contract) to send blood rushing out.” The factual question for this concept asks, “Which component of the circulatory system acts as a pump?” Answer: the heart.

For each of the 16 concepts, we also created an applied question through brainstorming sessions by asking ourselves the questions, “Why is this fact or component important to the circulatory system?” or “What would happen if this component was not functioning properly?” In most of these cases, the 16 applied questions reference the consequence of the factual relationship (described in the text) not holding true. For example, many applied questions require participants to predict outcomes given a certain component not functioning normally. The key principle for the applied questions is that participants must retrieve the necessary fact or facts from memory (presented previously in the text) and apply them in a new way. Importantly, the text only discusses the normally functioning circulatory system, and presents the material at the factual level, without much elaboration. Therefore, the applied questions are not presented explicitly in the text, but can be answered by processing and recombining the facts contained within the text. For the previous example, the concept of the heart acting as a pump, the applied question is, “Why doesn’t oxygen-rich blood flow directly from the lungs to the rest of the body?” Answer: Because blood requires a pump, the heart, to push it through the body.

2.3.3 Multiple Choice Answer Options

Each question, both factual and applied, required three (incorrect) answer options for the multiple choice format. The incorrect answer options were created based on common misconceptions about the circulatory system. Information on misconceptions was gathered through past research [e.g., 32] and pilot testing (common incorrect responses to the questions in short answer format). Once three answer options (in addition to the correct answer) were created for each of the factual and applied questions, additional pilot testing confirmed that the frequencies of responses for each of the three incorrect answer choices were not substantially different from each other. This method for creating the answer options was specifically done in an attempt to not lessen the effort required to answer a multiple choice item by using answer options that were unrelated or too easy for a participant to exclude as a possible answer.

2.4 Procedure

The experiment consisted of four portions (pretest, reading, retrieval practice, and posttest) within a single session delivered online through Amazon’s Mechanical Turk web service using the MoFaCTS online tutoring system (http://mofacts.optimallearning.org/) [27]. The entire experiment took an average of approximately 60 minutes for participants to complete. After obtaining informed consent, participants completed a pretest consisting of six factual questions. For each participant, half of the questions were randomly assigned to short answer format and the other half to multiple choice. These six questions were created from the text in the same way as those for retrieval practice, but did not overlap with the 16 concepts covered in retrieval practice to reduce the possibility of priming. No corrective feedback was given during the pretest.

1 Experimental materials are available upon request; please contact the first author.
Next, the participants were asked to read the Circulatory System text which was displayed on a single screen (with a scroll bar). For this portion, participants were instructed not to take notes while they read the text. Participants read at their own pace without a time limit. The average time spent reading was approximately seven minutes.

Following the reading portion, participants began retrieval practice. Each participant was randomly assigned to practice with either factual MC, applied MC, factual SA, or applied SA questions. Retrieval practice consisted of eight questions (each representing a different concept covered in the text), repeated four times each. These eight items were randomly selected for each participant from the list of 16 concepts. The order of the eight questions was randomized for each of the four “blocks” of repetition. Corrective feedback was given immediately after participants entered their responses. Correct responses allowed the participant to immediately move on to the next item; incorrect responses were followed by a review period of 10 seconds, during which the correct response was shown on the screen. This feedback procedure not only provided the correct answer for the participant to review, but also provided an incentive for participants to try their best, since correct answers allowed the participant to “skip” the mandatory 10-second review period. In other words, participants would quickly realize that random guessing or poor effort would only increase the length of the experiment.

The final portion of the experiment was the posttest, which was given after a delay of approximately one minute. During this delay the participants were instructed to complete a “current emotion” survey discussed below. The eight concepts studied during retrieval practice were included in the posttest, but each was randomly assigned to be tested in one of the four format/depth conditions. Each participant also answered an additional eight posttest items (two in each of the format/depth conditions) which reference the eight remaining concepts that were not randomly selected for retrieval practice. This allowed us to see how well each practice condition transferred to similar but previously untested material. Each of the 16 posttest questions were presented once, in random order, without corrective feedback.

At three different points in the experiment, participants responded to a set of six “current emotion” questions. The three time-points were: before the retrieval practice to obtain a baseline, immediately after retrieval practice to look for an effect of practice condition on affect, and immediately after posttest to determine if the change in format and depth at posttest had an adverse effect on affect. Specifically, participants were asked to rate on a scale of 1 (Strongly Disagree) to 5 (Strongly Agree) how much they agree with the statement, “Currently, I am feeling ______.” This question was asked six times, with a different affect provided in the blank. The six affects were: anxious, bored, confused, discouraged, frustrated, and unfocused/distracted. Demographic information was also collected at the conclusion of the experiment.

2.5 Scoring
All questions were scored immediately by the system and received a score of 1 or 0 (although this value was not explicitly displayed to the participant). MoFaCTS (the online drill-trial problem authoring and deployment platform we used) scored short answer items by matching words in the participants’ responses to key terms necessary to answer the question correctly. Pilot testing revealed common (acceptable) synonyms and alternative words that we incorporated into the system to allow for slight variation in what was considered a correct response. For example, the (complete) correct answer for the (factual) question, “Where is the heart located in relation to the lungs?” is “The heart is located between the lungs.” The system scored the responses to this item based on whether or not it contained the word “between” or “middle.” The use of regular expressions embedded in the MoFaCTS programming allowed for any of the following responses to be counted as correct: “between the lungs”, “the heart is between the lungs”, or “the heart’s in the middle of the lungs.” The regular expressions in the system also accounted for ordering when applicable; for example, ordering is essential for the (factual) question, “Which gas do the cells of the body require to function and which gas do they expel as waste?” Participants received corrective feedback (either “Correct” or “Incorrect. The correct response is ______”) after each item in the retrieval practice portion, but not during the pretest or posttest.

3. RESULTS AND DISCUSSION
3.1 Overall Performance
Before we discuss the results of mining our retrieval practice data, it may be helpful to review the broader results of the experiment. Table 1 provides an overview of the average scores for the practice (8 items with 4 trials each), the portion of the posttest containing the eight concepts previously practiced, each randomly assigned to one of the four format/depth conditions, (total of 8 trials), and the portion of the posttest which consisted of eight previously unpracticed concepts, each randomly assigned to one of the four format/depth conditions (total of 8 trials).

The average performances during retrieval practice, provided in Table 1, support the general ordering of performance we expected for each condition. Namely, the Factual MC condition was the least difficult, with the highest performance during practice, the Applied SA was the most difficult condition as indicated by the lowest performance during practice, and the Applied MC and Factual SA fall in between in terms of performance during practice. A between-subjects Analysis of Variance (ANOVA) indicated significant differences between the four conditions, $F(3,174) = 28.49, p < .001$. Post hoc pairwise comparisons indicate that the only two conditions that are not significantly different from each other are the Applied MC and Factual SA conditions ($p = .19$). All other conditions are significantly different from each other (all $p$’s < .05).

<table>
<thead>
<tr>
<th>Retrieval Practice Conditions</th>
<th>Average Practice Performance</th>
<th>Average Posttest Scores†</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Factual MC ($n = 46$)</td>
<td>.85 (.12)</td>
<td>.65 (.17)</td>
<td>.45 (.23)</td>
</tr>
<tr>
<td></td>
<td>Applied MC ($n = 42$)</td>
<td>.77 (.17)</td>
<td>.70 (.21)</td>
<td>.45 (.24)</td>
</tr>
<tr>
<td></td>
<td>Factual SA ($n = 47$)</td>
<td>.73 (.15)</td>
<td>.69 (.14)</td>
<td>.50 (.19)</td>
</tr>
<tr>
<td></td>
<td>Applied SA ($n = 43$)</td>
<td>.55 (.18)</td>
<td>.68 (.20)</td>
<td>.47 (.21)</td>
</tr>
</tbody>
</table>

Note: † collapsed across all format/depth posttest conditions. Standard deviations in parentheses.

Table 1 also displays posttest performance for each condition on the eight concepts they had been tested on during practice, as well as on eight concepts they had read about in the text, but had not actively practiced. Between-subjects ANOVA’s showed no
significant differences between conditions for performance on either posttest. Note that the drop in performance from practice to the practiced concepts posttest is due to the within-subjects nature of the posttest conditions. In other words, the eight concepts were only practiced in one condition, but were then randomly assigned to be tested in one of the four depth/format conditions in the posttest, meaning that participants had two items in the posttest of practiced concepts that were in a different format, two that were in a different depth, and two that were in a different format and depth. These different types of transfer in the posttest for the practiced and unpracticed concepts therefore resulted in lowered overall performance. Although not significant, we do see that the Factual MC condition was most affected by these transfer items for the posttest on practiced concepts.

While the ANOVA’s offer us a broad view of overall performance, in order to truly answer our research questions we will need a finer-grained analysis. Mining our data and creating a model of learning will give us a more in depth look at what is taking place during retrieval practice.

3.2 Modeling Retrieval Practice

A logistic mixed-effects regression was created to model learning during retrieval practice. Since retrieval practice conditions differed in the question depth and answer format factors according to the result above, this model is meant to dissect the differential learning caused by each type of question. The model is based on a Performance Factors Analysis (PFA) where performance is predicted on subsequent trials as a function of the performance on prior trials [26]. Unlike Additive Factors Modeling (AFM) [8], PFA captures prior performance by two parameters, differentiating the effect of prior incorrect (unsuccessful) and correct (successful) trials. We chose to use PFA to separate these components because it would allow us to look into the difference in predictive ability between successful versus unsuccessful prior retrievals. This comparison would indicate if the benefit of retrieval practice is dependent on successful retrieval, or if the mere attempt at retrieval (i.e., incorrect trials) also results in better performance.

Modeling the data included several iterations guided by our hypotheses concerning the effects of format, depth, and prior knowledge. We began with the basic components of a PFA model: two parameters to capture the count of prior correct and incorrect trials. We also included pretest score and a random effect of subject, all of which were significant.

We then added in features we suspected would affect performance based on the cognitive and educational research discussed above, namely, the format and depth of the practiced items. We used one parameter to capture the format of the current item and one to capture the depth of the current item. We also tried adding measures of response time (e.g., time spent reading the text prior to practice, average time spent on all previous trials, and average time spent on previous trials with the specific item, etc.) but none were significant in the model. Next, we added interactions between all factors that had proven significant at that point (e.g., count of prior correct by depth, count of prior incorrect by pretest, depth by format, etc.) Only two of these interactions were significant: count of prior correct by format and count of prior correct by depth, which were retained in the final model. Finally, several measures of affect were added to the model (i.e., the affective score).

The final additions to the model included measures of affect. Remember that our measure of affect consisted of six questions which each used a 5-point Likert-item (1- Strongly Disagree to 5- Strongly Agree) for participants to rate how much they agreed with the statement: “Currently I am feeling _____” for each of the six different affects (anxious, bored, confused, discouraged, frustrated, and unfocused/distracted). Ratings for each of these six affects were collected before and after retrieval practice (and after posttest, but that was not relevant to modeling the learning during practice). We tested the model using six parameters of the affects before practices, and then six parameters to capture the affect after practice. We decided to try to approximate participants affective states during practice by averaging the self-reported levels of affect reported before and after practice. It should be noted that there was not much change in affect from before to after retrieval practice, and each of the three measures (the “before” ratings, the “after” ratings, and the average of the two) performed similarly in the model. Confusion (averaged to capture affect during practice) was the only affect factor that improved the fit of the model. The last step was adding in interactions between this confusion measure and the count or prior correct and incorrect trials, of which only the latter was significant. The final model, summarized in Table 2, retained each of the parameters that achieved significance throughout our modeling process.

The final model had an $R^2$ of .359, with 5,696 total observations from 178 participants. The AIC was 4838.2, the BIC was 4904.6, and the Log Likelihood was 4818.2. Table 2 summarizes the fixed effects parameter values of the final model. Not included in Table 2 is the random effect of Participant ($SD = 0.669$). For the format and depth parameters, a value of 0 was assigned to the less difficult level (i.e., MC and Factual) and a value of 1 was assigned to the more difficult level of each factor (i.e., SA and Applied). For each of the parameters involving the count of prior correct or incorrect trials, the log of $(1 + \text{the prior count})$ was taken to account for diminishing marginal returns expected from the power law of practice [25]. Figure 1 also illustrates the fit of the model (left) to the participants’ data (right).

Ten runs of a 10-fold cross-validation revealed that the model retained validity when comparing the training folds ($R^2 = .336$) to the testing folds ($R^2 = .329$). The CV proportion (training folds $R^2$ divided by testing folds $R^2$) for the model indicated that 97.9% of the validity of the model was retained in the held out data.

Table 2. Summary of Fixed Effects for Logistic Regression Model Predicting Future Success

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Estimate</th>
<th>SE</th>
<th>Z-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>-0.11</td>
<td>.19</td>
<td>-5.56</td>
</tr>
<tr>
<td>Pretest</td>
<td>1.95</td>
<td>.30</td>
<td>6.50</td>
</tr>
<tr>
<td>Count of Prior Correct</td>
<td>1.82</td>
<td>.16</td>
<td>11.72</td>
</tr>
<tr>
<td>Count of Prior Incorrect</td>
<td>1.47</td>
<td>.15</td>
<td>9.88</td>
</tr>
<tr>
<td>Format</td>
<td>-1.22</td>
<td>.14</td>
<td>-8.93</td>
</tr>
<tr>
<td>Depth</td>
<td>-0.82</td>
<td>.13</td>
<td>-6.06</td>
</tr>
<tr>
<td>Prior Correct x Format</td>
<td>1.13</td>
<td>.19</td>
<td>5.93</td>
</tr>
<tr>
<td>Prior Correct x Depth</td>
<td>0.36†</td>
<td>.19</td>
<td>1.93</td>
</tr>
<tr>
<td>Prior Incorrect x Confusion</td>
<td>-0.18</td>
<td>.05</td>
<td>-3.78</td>
</tr>
</tbody>
</table>

Note: † $p < .05$; all other parameters are significant at the $p < .001$ level. For the Format and Depth parameters, MC and factual are coded as 0, and SA and applied are coded as 1, respectively.
3.3 Model Interpretation

One of the first things the data mining reveals is that correct retrieval (specifically recall) is important for learning. However, the current model also indicates a benefit from unsuccessful retrieval, although to a smaller degree. It is worth noting the model also shows a (lesser) benefit from unsuccessful trials. When comparing just the effect of prior correct and incorrect practice trials, it appears that they offer almost equivalent additions to the prediction/model (1.47 vs 1.82). However, the count of prior correct also interacts with the depth and with the format. For three out of the four practice conditions, these increase the predictive ability of previous successful practices. Therefore, taken all-together, there is much more of a positive effect of previous correct trials than incorrect trials. For example, in the Applied SA condition with one previous correct trial and one previous incorrect trial, successful practices is more than twice as impactful on future performance as previous unsuccessful practices when taking the interactions into account. This difference between the influence from previous correct versus incorrect trials is made even greater if the student has a higher level of confusion (as indicated by the negative estimate for the confusion*incorrect count parameter). This result adds to the building body of research that suggests it is successful retrieval, and not just the attempt to retrieve, that is beneficial to learning [20; 21; 29]. Thus, when it comes to supplying challenging questions for retrieval practice, we must be sure that the questions are at an appropriate difficulty-level for the student, so the student can be successful enough to gain from such practice.

Our model also shows how the format and depth of a practice item influence performance. First we see that the average performance for multiple choice practice is significantly higher than practice with short answer (as indicated by the overall performance of the multiple choice conditions during practice in Table 1 and the -1.22 estimate for short answer practice in Table 2 and), which indicates that multiple choice is the better option in terms of allowing for a higher percentage of successful practice. However, we also saw in the model above that there is more gained from successful short answer practice than is gained from successful multiple choice practice (the Prior Correct x Format parameter). This result are aligned with prior work which suggests that the short answer format may not be universally “better,” especially if students are not getting a sufficient amount of those questions correct [31]. Based on these results, it is reasonable to suggest that in order to schedule effective practice, students should be given questions that have a higher likelihood of being answering correctly. If we assume that for the most part, students have a lower level of prior knowledge at the beginning of practice/learning a topic, multiple choice item may permit learning by boosting success. However, since successful short answer practice offers more of a benefit (than multiple choice), it seems that students should eventually transition into short answer practice as they become more proficient. In other words, practice should begin with the less effortful item-type and transition to the more effortful (and more beneficial) item once students reach some level of mastery.

The same may be said for practice with the deeper applied items, over the more text-based factual questions, in that students will get the factual items correct more often, but there is more gained from successful applied practice than from successful factual practice. Again, students might benefit most from beginning with the easier depth (factual/ text-based) and finishing retrieval practice with more difficult, applied questions. The goal it seems, should be to get students to a point where they can get many successful retrieval attempts with SA and/or applied items. This suggestion aligns with ideas in several areas of education research including scaffolding [17], zone of proximal development, and concreteness fading [34]. Determining the optimal level of mastery is an important component though, since increased redundancy during learning (repeated practice of known information) has been shown to offer decreasing marginal returns [9; 28]. Our model also illustrates the importance of taking prior knowledge into account when designing tutoring systems and practice schedules. Some students might be able to begin right away with more difficult items (e.g., applied short answer) and others would benefit from beginning practice with factual multiple choice questions and progress from there.

3.3.1 Affect in the Model

The work concerning affect in the current study is exploratory in nature and was meant to give us an indication of which affective states might be the most important to investigate further in future experiments. Our measure of affective states indicated that the most influential affect was confusion. The interaction between the count
of prior unsuccessful trials and self-reported confusion level in our model shows that when a learner answers more questions incorrectly, higher confusion predicts a much larger negative effect than if a learner has higher confusion but is still having mostly successful practice. This preliminary result appears to align with previous findings which suggest that confusion can be an important component during learning, and is beneficial when students identify that confusion and work to clarify it (i.e. start to produce correct responses), but detrimental when the confusion is overwhelming or the student fails to remedy it [12].

Unlike previous work by Baker, et al., [2] we did not find any significant impact of frustration or boredom (nor for the other affective states we asked participants about: anxiety, discouragement, and distractedness). As the current work was meant to serve only as an exploration of affect during retrieval practice, this is an area that we may investigate further in the future. In future work we may implement pop-up/immediate questions concerning the participant’s current affective, or specifically their level of confusion, after more than one incorrect response to answer items, is more beneficial than difficulty created through the retrieval hypothesis, that successful trials benefit for successful retrieval of short answer over multiple choice items, [3,4].

3.4 General Conclusions
Our model of performance during retrieval practice indicates a benefit for successful retrieval of short answer over multiple choice items. Likewise, there is a benefit from successful retrieval of applied items over factual items which supports the effortful retrieval hypothesis, that successful trials with more difficult items are better than success on less difficult items. Our hypothesis that the difficulty of multiple choice items could be increased (and equated with difficulty of factual short answer items) by asking applied questions, could potentially be supported by the non-significant difference in practice performances, although more analyses will be necessary before making this conclusion. However, format appears to be a more powerful predictor of future success than depth. This may suggest that the difficulty of retrieving information from memory created from less cues (short answer items), is more beneficial than difficulty created through the effortful processing and reasoning with retrieved information (applied items). We recognize that the construct of retrieval effort could be considered too broad of an explanation for our results. While retrieval effort may not capture all the nuances involved in understanding retrieval, we believe it offers a parsimonious general framework under which several mechanisms are captured. Understanding the role that effort plays in retrieval practice will benefit from future work that investigates the differences in more fine-grained mechanisms such as individual difference in strategy use and/or cognitive processes involved in practice with each question type.

4. ACKNOWLEDGMENTS
Our thanks to the University of Memphis’s Institute for Intelligent Systems for providing the resources and support necessary to conduct this research.

5. REFERENCES


ABSTRACT

While Educational Data Mining research has traditionally emphasized the practical aspects of learner modeling, such as predictive modeling, estimating students' knowledge, and informing adaptive instruction, in the current study, we argue that Educational Data Mining can also be used to test and improve our fundamental theories of human learning. Using the Apprentice Learner architecture, a computational theory of learning capable of simulating human behavior in interactive learning environments, we generate two models that embody alternative theories of human learning: (1) that humans perfectly recall previous training during learning and (2) that humans only recall a limited window of experience. We evaluate which of these models is better supported by data from two fractions tutoring systems. In general, we find that the model with a complete memory better fits the data than a model recalling only the previous training experience (data-driven theory development). Additionally, we demonstrate that both models are able to predict student performances, as well as, reproduce the main effects of an experimental paradigm without being trained on student data (theory-driven prediction). These results demonstrate how the Apprentice Learner architecture can be used to close the loop between learning theory and educational data.

1. INTRODUCTION

One branch of Educational Data Mining (EDM) research leverages data to improve our theoretical understanding of how people learn [28, 3]. Analogous to how data from the Large Hadron Collider can be used to gain insights into physical laws, educational data can be used to provide insights into the unobservable mechanisms underlying student learning. Surprisingly, little EDM research has explored this direction, rather, the main trends in research center on how statistical models can be used to perform latent knowledge estimation and domain-structure discovery (i.e., knowledge component discovery) [3]. While these research directions are important, we argue that the availability of educational data makes the EDM community well poised to contribute substantially towards our theoretical understanding of human learning.

Although many of the widely used predictive models of learning, e.g., Bayesian Knowledge Tracing [5], and Additive Factors Model [4], rely on existing theories of human learning, such as the power law of practice [22], researchers rarely apply these models to educational data with the aim of improving the underlying theory of learning. Further, there are a number of barriers to using educational data for this purpose. First, many EDM models are only loose approximations of the theories they are based on. For example, the Additive Factors Model predicts that improvements in human performance will follow a single logistic function, whereas the power law of practice states that the improvements should follow a power function [6]. Second, EDM models do not reflect the current state of learning theory. For example, recent studies of skill acquisition actually suggest that improvements should follow three distinct power functions, one for each phase of cognitive skill acquisition [30], rather than a single logistic function. This disconnect between theories and models makes it difficult to draw inferences about the underlying theories given the fit of models to data. By more tightly connecting our EDM models to theory, we can leverage educational data to improve our understanding of the mechanisms behind human learning and, in turn, use these theories to improve our abilities to predict student behavior.
To more tightly link a theory to models, researchers can develop a computational theory [17, 21]. Unlike a theory that only specifies the abstract relationships between constructs (e.g., that an increase in spatial skills leads to an increase in learning with graphical representations [26]), a computational theory represents a complete description of the mechanisms that produce observed phenomena. Within this paradigm, a model presents as a specific algorithmic implementation of these mechanisms that can be executed to simulate behavior, which then can be compared with observed behavior in order to test both the model and the underlying theory. A key component of this approach is not to explain or “fit” a relationship in observed data, but rather, to predict that a relationship will be present before any data is observed. Figure 1 shows the iterative relationship between theories, models, and behaviors. We argue that this approach complements existing approaches in the EDM literature.

In the current work, we present the Apprentice Learner architecture, a computational theory of learning in interactive learning environments, such as tutoring systems. Unlike prior models of student performance, such as Additive Factors Model and its variants, Apprentice Learner models seek to explain the mechanism students use to acquire new knowledge from instruction. This mechanical description allows us to simulate learner behavior within an instructional environment and use these simulations to predict human behavior. Rather than arriving at a general conclusions like students learn differently from positive and negative feedback this approach lets us explore possible explanations for the mechanisms driving these results. In presenting this computational theory we make two claims:

1. The Apprentice Learner architecture can be used to predict student behavior and experimental results before collecting any student data (purely theory-driven prediction).
2. The Apprentice Learner architecture can leverage data to improve learning theory through the creation and testing of different models of learning.

To support these claims, we explore different assumptions about memory and its effect on human learning in intelligent tutoring systems. We leverage the the Apprentice Learner architecture to instantiate two models of human learning, one that hypothesizes perfect memory and another that assumes a more limited window of memory. We apply these models in two different fractions tutoring systems. In both cases, we generate datasets of simulated learner behavior that have high agreement with the patterns of behavior observed in human students. Additionally, we show that our models reproduce the main effects of a problem sequencing experiment without first being fit to student data. In general, we find that the model with perfect memory better fits the fractions data than the model with limited memory; these findings provide an initial demonstration of how our computational theory can be refined in response to data.

In the following sections, we first present the Apprentice Learner architecture and describe the theoretical commitments that it makes. Next, we describe our overarching simulation approach, the particular computational models that we investigate, and the results of our simulation studies in (1) fraction addition and (2) fraction arithmetic. Finally, we discuss the implications of our results and directions for future work.

2. THE PROPOSED ARCHITECTURE

In 2006, VanLehn published his seminal paper describing the step-level behavior of tutoring systems [31]. Although not commonly cited within the EDM literature, VanLehn’s description of the general two-loop structure of tutoring systems (i.e., an inner loop for step-level feedback and an outer loop for problem selection) has direct relevance to many recent advances in EDM research. For example, researchers have used knowledge component discovery to create a better understanding of domain tasks [4, 13], so that the inner loop feedback can be improved. Other researchers have used latent knowledge estimation to improve outer loop instructional policies [27]. While VanLehn’s theory promotes common ground between similar thrusts of work in EDM, it can only serve as half the picture of a computational theory of the tutoring process.

The Apprentice Learner architecture, shown in Figure 2, is a computational theory of human learning that aligns with the step-level interactions described by VanLehn. The theory embodied in the Apprentice Learner architecture states that students acquire skills by interactively solving problems in a tutored paradigm, receiving correctness feedback on their actions. In the event that the student does not know how to proceed, they can request a hint from the tutor, which provides the student with a demonstration of how to take the next problem-solving step.

The Apprentice Learner architecture uses a base of prior knowledge to induce new skills from its observed demonstrations and feedback. The first kind of knowledge consists of functions for manipulating data (e.g., adding two values, appending two strings together, etc.). The second kind of knowledge consists of features for recognizing different elements in the interface (e.g., recognizing numbers, mathematical symbols, etc.). Depending on the domain, different kinds of background knowledge may be appropriate. For example, Apprentice Learner models in equation solving might have features for recognizing polynomials, whereas models in stoichiometry might have different features for recognizing chemical symbols.

The Apprentice Learner architecture posits three learning mechanisms to induce new skills from prior knowledge and observed demonstrations and feedback. When given a demonstration, the how learning mechanism uses function knowledge to search for a sequence of functions that can explain the observed demonstration. After discovering a function sequence, the when learning mechanism acquires general perceptual patterns for recognizing the elements used in the discovered sequence. Finally, the when learning mechanism uses the tutor state, augmented with feature knowledge, to identify the conditions under which the discovered sequence should be executed. The combination of the components discovered by how, where, and when learning mechanisms constitutes a skill. Apprentice learners apply learned skills in subsequent problem solving.
In order to apply learned skills, the Apprentice Learner architecture posits that learners use a basic Recognize-Act cycle [32]. When presented with a problem, learners first query their skill knowledge to determine if any known skills are applicable. If an applicable skill is found, the learner executes it. The learner passes correctness feedback on the resulting action to its when learning mechanism, which uses the feedback to refine the conditions under which the skill can be executed. In the event that no skills are applicable, the learner requests a demonstration that is passed to the how, where, and when learners to produce a new skill.

Given the computational theory described by our architecture and our data-driven theory development approach (see Figure 1), our goal is to develop a theory that is consistent with available educational datasets, such as those found in DataShop [7] and other similar repositories. To pursue this goal, we propose a research program wherein different models of human learning are generated within the framework of the Apprentice Learner architecture, i.e., specific algorithms are implemented for each of the components of the architecture. These Apprentice Learner models can then be connected to the same intelligent tutoring systems that generated the data found on Datashop. Next, the behavior of these models can be compared to human behavior. Based on the differences between the models and humans, we can revise our theory (e.g., replacing a perfect memory of previous demonstrations and feedback with a memory that only recalls a window of experience), generate new models, and then simulate the revised models to determine if better agreement between models and human behavior can be demonstrated.

3. SIMULATION STUDIES

We make two key claims about the Apprentice Learner architecture: (1) it can be used to predict student behavior without data and (2) it can be used to improve theory by facilitating the exploration of different models. To demonstrate the potential of the architecture and to support our key claims, we conducted simulation studies with two tutoring systems in the domain of fractions [33, 14, 24]. For these simulations, we created an initial model of human learning by implementing each of the components of the Apprentice Learner architecture in computer code. This model was given two features, isPlusSign and isMultSign, which can be used to determine if a string is a plus or multiply sign (i.e., + or \(\times\)). It was also given six functions: Add(X,Y), Subtract(X,Y), Multiply(X,Y), Divide(X,Y), CopyPasteString(X), and GenerateCheckMark(). The Add, Subtract, Multiply, and Divide functions returned the result of applying their respective arithmetic operations to their arguments. The CopyPasteString function returns a copy of the string that is passed to it. Finally, the GenerateCheckMark takes no arguments and returns a check mark that can be used to fill checkboxes in the tutor interface. This prior feature and function knowledge represents the basic interface and arithmetic knowledge that students would be expected to know before using a fractions tutor.

In this initial model, the skill knowledge acquired from the three learning mechanisms is stored in the form of production rules (i.e., IF-THEN rules). The perceptual patterns learned from the where learner and the conditions acquired by the when learner constitute the IF part of the rule. The function sequence discovered by the how learner constitutes the THEN part of the rule. An example of a human-readable version of a production rule discovered by one of our models might be:

\[\text{IF} \quad \text{there are two fractions with denominators}\]
tutors and a sign between them (i.e., the perceptual pattern is present) **AND** the sign is a plus sign and the denominators are equal (i.e., the conditions are satisfied) **THEN** copy one of the denominator values and put the result in the answer denominator box (i.e., perform the function sequence). During problem solving, the models check if they have any applicable production rules (i.e., skills) and if a match is found, then they take the prescribed action.

This initial model, which we refer to as the full-memory model (also the SimStudent model in previous work), has been used to model ordering effects [10], as a teachable agent [18], and for authoring cognitive models [16, 12]. In these previous studies, the full-memory model has been found to regularly outperform human students. One hypothesis is that this model outperforms human students because it revises its skill knowledge using a complete memory of all previous training examples [15]. To explore this hypothesis, we created a second model that duplicates our initial model, with the exception that it only recalls the previous training example during skill learning. This model, which we refer to as the one-back-memory model, instantiates an extreme version of the hypothesis that learners only recall a limited amount of their past experience during learning.

### 3.1 Data
To test the full-memory and one-back-memory models, we use data from two intelligent tutoring systems available on DataShop. In both tutors, students were asked to solve fraction arithmetic problems using a variation of the interface shown in Figure 3. The first dataset came from the control condition of a fraction addition study [33]. The dataset consisted of 24 students solving 20 fraction addition problems. The tutoring system used in this dataset omitted the “I need to convert these fractions before solving” checkbox and required students to convert fractions to common denominators, even if this meant copying fractions that already had the same denominators. Additionally, the tutor allowed students to use multiple approaches to find a common denominator; they could either multiply the denominators or compute the least common denominator. To allow our models to use this second approach, we added the LeastCommonMultiple(X,Y) function to the prior knowledge of both models, under the assumption that students utilizing this approach know how to compute the least common multiple.

The second dataset came from an experiment testing whether blocking or interleaving different types of fraction arithmetic problems was better for learning [24]. This dataset contains 79 students solving 24 fraction addition problems (10 with same denominators and 14 with different denominators) and 24 fraction multiplication problems. The tutor used in this study required students to check the “I need to convert these fractions before solving” box before making the fields necessary for converting visible. Additionally, on fraction addition problems with different denominators, students were only allowed to compute common denominators by multiplying denominators. Thus, the LeastCommonMultiple(X,Y) function was not included in the models for this dataset.

The experimental manipulation of the second datasets divided students into two conditions, blocked and interleaved. The students in the blocked condition received three blocks of problems: fraction addition problems with same denominators, then fraction addition problems with different denominators, and then fraction multiplication problems. The order of the problems within each block was randomized for each student. In contrast, the students in the interleaved condition received a random ordering of all problems. This experiment showed that students in the blocked condition have a lower overall error rate than students in the interleaved condition. Additionally, the error rates of students in the blocked condition increased when transitioning between different types of problems.

### 3.2 Method
For each dataset, we tested our full-memory and one-back-memory models of learning by creating instances of each model for each student and connecting these instances to the appropriate tutoring systems. The tutoring systems then tutored the instances through the same order of problems that the respective human students received. In each dataset, we compared the first attempt correctness on each step between the two models and their respective humans. For each model, we computed how often the first attempt correctness agreed with the respective human’s first attempt correctness, i.e., accuracy, to quantitatively measure the agreement between model and educational data. We report the mean accuracy and its accompanying 95% confidence interval (95% CI) for each model. Next, for each dataset we plotted overall learning curves comparing the first-attempt performance of the humans to each of the two models. For these learning curves, we used a knowledge component model that labeled each step as exercising a skill corresponding to the field that was updated in the interface. These learning curve graphs demonstrate how the Apprentice Learner architecture can be used to generate theory-driven learning curve predictions. Because each model instance has the same prior knowledge, our simulation studies do not take into account the individual differences in students’ prior knowledge. To determine if taking into account student-level effects impacts which model better fits the data, we fit a random-effects logistic regression model with a fixed effect for the model prediction and random effect for the student. We report the Akaike information criterion (AIC) and Bayesian information criterion (BIC) scores to determine which of our two models better fits the data in each case. Note, AIC and BIC values on one dataset are not comparable to the AIC and BIC values on another dataset, they can only be used to...
rank model fits on the same dataset. For a given dataset, lower values of AIC and BIC are better, and a difference of more than 3 in either measure is usually viewed as strong evidence to prefer one model over another.

### 3.3 Fraction Addition Results

After simulating the the 24 students in the fraction addition dataset, we found both models were significantly predictive of students’ correctness on the 2,432 first attempts ($p < 0.01$ via a $\chi^2$ test). The full-memory model correctly predicted 74.05% (95% CI : 72.26, 75.79) of first attempts, whereas the one-back-memory model correctly predicted only 70.93% (95% CI : 69.08, 72.73) of first attempts. This significant difference in accuracy ($p < 0.01$ via McNemar’s test) suggests that the full-memory model more closely agrees with the fraction addition data than the one-back-memory model, when not taking into account differences in students prior knowledge.

Next, we plotted the learning curves comparing both models’ performance to the human performance, see Figure 4. The opportunity counts for these learning curves were determined by how many times each student had practiced filling in the relevant interface field (each field is roughly analogous to the skill used to update that field). Both simulated models initially start off without any skills, so their error rate is 100% on the first step. However, the models quickly converge to human-level performance. Although the full-memory model achieves a lower overall error, the one-back memory model appears to have variation that is more equally distributed around the human performance.

To test which model best fits when taking the differences between students’ prior knowledge into account, we fit two mixed-effect logistic regression models that had a single fixed effect for the respective simulation prediction (full-memory or one-back-memory) and a random effect for student. We found that the one-back-memory model better fit the student data (AIC=1727, BIC=1744) than that full-memory model (AIC=1754, BIC=1772), suggesting that students in the fraction addition dataset have differences in their overall performance that might correspond to differences in prior knowledge. Further, these results suggest that the one-back memory model better fits student performance when taking these differences into account.

### 3.4 Fraction Arithmetic Results

Similar to the previous dataset, we found both models were significantly predictive of the 79 students’ 18,589 first attempts ($p < 0.01$ via a $\chi^2$ test). We also found that the full-memory model (Accuracy : 84.04%, 95% CI : 83.5, 84.56) was more predictive of students’ first attempts than the one-back-memory model (Accuracy : 80.24%, 95% CI : 79.66, 80.81). Similar to our previous fraction addition results, this significant difference in accuracy ($p < 0.01$ via McNemar’s test) suggests that the full-memory model more closely agrees with the fraction arithmetic data than the one-back-memory model, when not taking into account differences in students prior knowledge.

Figure 5 shows the learning curves comparing the performance of the two models to the human data. Similar to the fraction addition dataset, the opportunity counts for these learning curves were determined by how many times each student had practice filling in the relevant interface field (again, fields are roughly analogous to the skills used to update them). However, in this dataset we plotted separate learning curves for students in the two experimental conditions, blocked and interleaved.

Similar to the fraction addition learning curves, the full-memory and one-back-memory models initially start off with an error rate of 100% on their first steps and quickly converge to human-level performance. However, in this dataset, we can see that both models seem to emulate key differences in the two conditions. First, the human students in the blocked condition have lower error than those in the interleaved condition ($z = -6.136, p < 0.01$ via a logistic regression). Both the full-memory ($z = -9.598, p < 0.01$) and the one-back-memory ($z = -4.626, p < 0.01$) models correctly predict this main effect of condition. Second, the human students in the interleaved condition slowly converge to asymptotic performance, whereas the human students in the blocked condition achieve lower initial error but then have drastic increases in error when transitioning between problem types (e.g., around opportunity 12). The simulated data from both models appears to mirror these effects. While both models experience a spike in error around opportunity 25 when transitioning to multiply problems, the human students, surprisingly, do not show a similar increase. This difference might be explained by the fact that the human students have prior experience multiplying numbers, and fraction multiplication is arguably easier than fraction addition with different denominators (i.e., students have to use multiplication to compute common denominators). In contrast, both the full-memory and one-back-memory models have no experience with multiplication prior to opportunity 25, so they have a 100% initial error on the first multiplication step. This suggests that future work is needed to explore how to populate models with initial training experiences (e.g., teaching the model to do whole-number multiplication before fraction multiplication).
Finally, we again fit two mixed-effects logistic regression models to determine if taking individual student differences into account would change which of the two models better fit the data. In contrast to the fraction addition results, we found that the full-memory model better fit the student data (AIC=10849, BIC=10872) than that one-back-memory model (AIC=11013, BIC=11036). These results show that, for fraction arithmetic, the full-memory model better fits the student data regardless of whether or not overall student differences are taken into account.

4. GENERAL DISCUSSION

We argue that our simulation studies in fraction addition and fraction arithmetic provide strong evidence in support of our two key claims about the Apprentice Learner architecture. First, our analysis shows that the behavior generated by both models agrees with the human behavior in both fractions datasets; i.e., the full-memory model, which fits best, achieves 75% agreement in the fraction addition dataset and 84% agreement in the fraction arithmetic dataset. Furthermore, we show that both of the models predict the main experimental effect for the fraction arithmetic dataset; i.e., both models correctly predict that the overall performance in interleaved condition will be better than the overall performance in interleaved condition. To our knowledge, these two results are the first example in the EDM literature of how student performance can be precisely predicted in a completely theory-driven way without having to fit the models to the student data first.

Although our models have a reasonably high agreement with the student data, there are still some key differences between the models and the humans. In particular, the models always have 100% first-attempt error on novel skills. While these exaggerated error rates might be useful for detecting transitions between skills (e.g., when using learning curve analysis to develop knowledge-component models [5]), they also suggest an opportunity to improve our underlying theory and models. In future studies we should explore approaches for initializing both prior knowledge (e.g., using students’ pretests to choose prior features and functions) and skill knowledge (e.g., pretraining models in a whole-number arithmetic tutor).

Our second key claim was that the Apprentice Learner architecture can be used to improve our underlying theory of human learning using educational data. We argue that our simulation results provide strong evidence supporting this claim. In particular, we tested two different models that operationalize two alternative theories of human learning: the full-memory model, which posits that humans have perfect recall of prior demonstrations and feedback when learning skills, and the one-back-memory model, which is an extreme version of the theory that humans only recall a limited window of prior demonstrations and feedback during skill learning. In our analysis, we showed that the full-memory model better fits both fractions datasets, suggesting that it is a better model of human learning. Next, we used a mixed-effects logistic regression analysis to take into account student differences. Using this approach, we showed that the one-back-memory model better fit on the fraction addition dataset and the full-memory model better fit on the fraction arithmetic dataset.

In general, these results suggest that the full-memory model better fits the fractions datasets than the one-back-memory model (in three out of four cases). However, our results leave open the possibility that, when taking into account overall student differences, a hybrid model might be best (e.g., an n-back model). Further, the full-memory model best fits the educational data, but seems to have better asymptotic performance than the human students. The original inspiration for the one-back-memory model was to decrease this asymptotic performance to bring it into closer alignment with the human performance, but our results suggest that we should consider alternative approaches for decreasing performance.

Figure 5: The fraction arithmetic learning curves for the human students, the full-memory model, and the one-back-memory model. The left graph shows the learning curves for the blocked condition and the right graph shows the learning curves for the interleaved condition. The spikes in error rate in the blocked condition occur when students transition from fractions with same denominators to fractions with different denominators (opportunity 12) and to fraction multiplication (opportunity 25).
One possibility would be to replace the when learner with an incremental machine learning algorithm, such as TRESTLE [15]. This approach would let apprentice learners leverage existing theories of interference effects [2] to improve their fit with educational data. In summary, our simulation studies provide strong evidence to support our claims that the Apprentice Learner architecture can be used perform theory-driven prediction and to improve theory based on differences between model and human behavior.

5. FUTURE WORK
The results of our studies have been encouraging, however, we do not wish to leave the impression that the Apprentice Learner architecture is a complete computational theory of learning. Instead, we present the theory as an initial framework that is flexible enough to support new hypotheses about learning. In future work, we plan to explore several variations of the current theoretical structure and invite the community to extend the theory to explain phenomena in their own work.

One affordance of the Apprentice Learner architecture is that it facilitates a search among alternative theories and models. Not unlike existing techniques for searching the space of domain models [4], a search among alternative Apprentice Learner models would let us explore several hypotheses of human learning. For example, it is questionable whether how, where, and when are the correct combination of internal learning mechanisms. It may be that the FOIL algorithm, currently used for when learning, could be used to model both the where and the when learning. This would suggest that the current distinction between where and when learning is artificial and that a single mechanism might produce more human-like simulated data. Alternatively, it could be argued that the architecture is biased by having features provided as prior knowledge, rather than learning features from experience. This argument implies that some mechanism for acquiring new features, effectively a what learner, could be included in the architecture [11]. Beyond adding or merging learning mechanisms each individual mechanism could be represented by several underlying algorithms. For example, our implementation of the Version Space algorithm conducts a specific-to-general search for perceptual patterns, but another possible variation would be to conduct a general-to-specific search. Exploring all of these possibilities could be framed as a search task over different parametrizations of the architecture for models that generate the most human-like simulation data.

In the current work, we compare model and human error rates, but the Apprentice Learner architecture allows for finer-grained evaluation. Rather than compare simulated and human learners on whether they performed a step correctly, we could compare learners in terms of their literal response on a step. This opens up the ability to evaluate theories of student misconceptions and how they might affect the particular responses students make [19]. Similarly, in this study we only compared performance on first step attempts, because this is a common convention in EDM, but the high-fidelity simulation data can be used to examine learner behavior beyond the first attempt. Ultimately a unified theory of apprentice learning should account for all of the behaviors learners exhibit on their path to mastery.

As we have stated previously, we view the current state of the Apprentice Learner architecture as incomplete. There are several aspects of learning that the model does not currently account for, such as the effects of delayed feedback [29], the impacts of metacognition [1], and the behavior of collaborative learners [23]. Crucially, however, the theory is not fundamentally incompatible with these ideas. For example, a reinforcement learning paradigm could be employed to back-propagate correctness from delayed feedback. The role of metacognition could be accounted for with a more nuanced variation of the recognize-act cycle that takes into account metacognitive decisions. Finally, instantiating multiple Apprentice Learner models within the same environment and allowing them to generate demonstrations for each other could serve as an initial computational model of collaborative learning. These are just a few examples of how the structure of the architecture can be augmented to incorporate and test additional learning theories.

Finally, in future work we would like to explore how the theoretical tenets of our architecture align with those made by other architectures, such as ACT-R or SOAR [8]. These architectures, which primarily focus on problem solving, have mechanisms for learning skill conditions and for compiling commonly executed sequences of skills into macro-skills. It would be interesting to investigate the extent to which these learning mechanisms align with the when (condition) and how (function sequence) learning mechanisms of the Apprentice Learner architecture. By investigating how these computational theories might be aligned, we hope to provide for learning science, and more generally cognitive science, the kinds of unified theories that have been so successful in physics and other hard sciences.

6. CONCLUSIONS
In this paper, we have taken the first steps toward a complete computational theory of learning in interactive environments, such as tutoring systems. Not only do we believe that EDM is capable of improving our fundamental theories of learning, but that is uniquely positioned to do so. Using a computational theory approach, it is possible for every tutored learning dataset in the canon of EDM to test and advance learning theories. We hope that other EDM researchers will also see the potential of the Apprentice Learner architecture and the computational theory paradigm, and we look forward to working together to further develop our collective understanding of human learning.

7. ACKNOWLEDGEMENTS
We thank Peggy Tenison for her feedback on earlier versions of this work. We used the “Grounded Feedback Fraction Addition Tutor” and “Fraction Addition and Multiplication” datasets accessed via DataShop (pslcdatashop.org). This work was supported in part by the Department of Education (#R305B000023) and by the National Science Foundation (#SBE-0836012).

8. REFERENCES


ABSTRACT
Effective mining of data from online submission systems offers the potential to improve educational outcomes by identifying student habits and behaviours and their relationship with levels of achievement. In particular, it may assist in identifying students at risk of performing poorly, allowing for early intervention. In this paper we investigate different methods of following the development of student behaviour throughout the semester using online submission system data, and different approaches to analysing this development. We demonstrate the application of these methods to data from a junior computer science course (N=494) and discuss their usefulness in understanding the common behavioural strategies of students in this course and how these develop over time. Finally, we draw links between behaviour in weekly coding tasks and student performance in the final exam and discuss whether these methods could be applicable midway through the semester.

Keywords
Clustering student behaviour; autograding system; assessment and feedback.

1. INTRODUCTION
Autograding submission systems are valuable tools in a modern teaching environment. By automatically assessing a student’s submission, feedback can be returned to the student immediately without increasing the burden of marking for the teacher. Students are empowered to repeatedly improve their submission before a final deadline. However, such systems are only likely to improve the student’s learning experience if the student allocates time to use feedback for subsequent submissions.

Teachers know from observation that students adopt a range of approaches to learning exercises, especially when outside the classroom. At one extreme, an ideal student will attempt an exercise immediately, and make increasingly better submissions based upon the feedback received. At the other extreme a student may make their first attempt just prior to the submission deadline, leaving no opportunity to improve or even make a decent first attempt. These behaviours, and many in between, may be due to deeply ingrained habits or external factors such as other time commitments. Using online submission systems in our teaching provide us with the opportunity to exploit the historical data of students’ attempts. In this work, we investigated techniques of identifying and following the development of student behaviour over the semester, with specific focus on the application of these techniques to a junior computer science course. We were interested in the most common behaviours of students, whether these behaviours changed over time, and relationships between these behaviours and final exam outcomes. We were also interested in how applicable these methods were midway through the semester.

This paper is structured as follows. We first give an overview of the related work on the use of autograding systems and on mining student behaviour in these systems. Section 3 explains the context in which our data was captured. Section 4 is the main part of the paper: it presents our clustering-based approach to detecting and tracking students’ behaviours. We finally conclude with a discussion on these different approaches.

2. RELATED WORK
The use of autograding systems in computer science courses have been reported in [1-6], with the majority of studies focusing on analysing the effectiveness of the autograding systems as opposed to understanding student behaviours. Sherman et al. [1] introduced Bottlenose, an autograding system used in a first year programming course in C, and compared the student behaviour on the same assignments when using Bottlenose and when not using it. The results showed that the number of submissions per student per assignment was significantly higher when using the autograding system, which was attributed to students making use of the feedback to improve their programs. Enström et al. [2] developed Kattis, an automated assessment system used at KTH in Sweden for teaching programming and algorithms courses. The use of Kattis resulted in improved student motivation (increased number of submissions) and also in higher student satisfaction in the course evaluation survey. The autograding system Autolab [3] was developed at Carnegie Mellon University and used in a first year programming course in C. Its real-time scoreboard, which shows the class performance on the assessment task, was found to create a healthy competition encouraging students to improve their assignments, and do this quicker.

There has also been some recent work on mining log data from autograding systems [4-6]. Gramoli et al. [4] analysed the impact of autograding and instant feedback using the system PASTA in various computer science courses, from first to fourth year. They found that the instant feedback was beneficial not only for courses focusing on programming but also for courses that use programming as a tool to solve subject specific problems. The relation between the student performance and the chosen programming language and the time when the students start and finish their assignment submissions was also studied. Koprinska et al. [6] investigated whether students at risk of failing in a first year
programming course can be detected early in the semester, using information from three sources: the autograding system PASTA, a discussion board and assessment marks. They built a decision tree that was able to achieve 87% accuracy in predicting the exam mark from information available in the middle of the semester. It was also shown that using the information from the autograding system improved the accuracy, compared to only using the assessment marks. In [5], data from the same sources was used to define the characteristics of high, average- and low-performing students and predict their performance.

More broadly, the related work also includes mining log data from student submissions in computer science courses. Perera et al. [7] analysed behavioural data from online group collaboration logs in a software development project. The goal was to identify patterns and behaviours associated with positive and negative outcomes. Clustering was applied to find similar students and similar teams, and sequential pattern mining was used to extract sequences of frequent events. Student behavioural data from a high school computer science MOOC was analysed by Tomkins et al. [8]. They characterised the performance of high and low achieving students based on the student behaviour in the course and discussion board, and built a predictive model using support vector machines to predict if a student will pass or fail an exam, conducted after the course has finished.

In this paper we extend the previous work on mining log data from autograding systems in computer science courses. Our goal is to study the evolution of student behaviour during the semester, with a view that this could assist in early intervention in future course offerings or provide guidance for course restructuring. We propose different clustering methods and demonstrate their application in the context of a large first year computer science course. We discuss the effectiveness of these techniques for extracting and understanding behavioural patterns, and how these patterns develop over time.

3. DATA
PASTA is an autograding system for computer programming courses developed in our school [9]. Students submit their solution (programming code) to an assessment task. Then PASTA checks this solution by running a set of tests designed by the teacher and provides immediate feedback to the student about the passed and failed tests. Students can then correct their mistakes and resubmit the solution until all tests are passed. PASTA can be configured in different ways - the number of allowed attempts can be limited or unlimited, some tests can be hidden (i.e. not available for immediate feedback, only available after the deadline) and teachers can also add manual comments to complement the automatic feedback. It supports several languages (e.g. Java, C, C++, Python and Matlab) and has been used for various courses – introductory programming, data structures, algorithms, formal languages, artificial intelligence, databases and networks.

PASTA has received positive feedback from students due to the instant feedback and multiple attempts features. Its use has resulted in better student engagement, and also transparent and fair marking as the same tests are used for all students. For each student and task, the PASTA data contains: all submission attempts, the tests that were passed and failed, the time stamps and the mark obtained.

The data used in this paper comes from a junior unit of study on data structures [10], which ran in Semester 2 of 2015 with 494 students enrolled. Students were using PASTA on a weekly basis to submit exercises, over a period of 11 weeks. The exercises were made available just after the lecture related to the topic (say Hashing) and constitute the core material of the tutorials (2 hour computer-based practical sessions, with a ratio of one teacher to 20 students). Each week, one exercise was flagged for assessment and was due the following week, i.e. 12 days after release. The number of attempts allowed was unlimited.

4. ANALYSIS OF STUDENT BEHAVIOUR
There are many ways students work towards their weekly exercises and use PASTA. For instance, students may start early and submit several attempts until their submission is 100% successful; some may start late and have time to submit only a half-done attempt; others may not submit anything at all; and so on. Our approach to follow students’ behaviour on their weekly work is to first cluster behaviours on all submissions, for all students (section 4.1). Then we explore several ways of tracking students’ behaviour during the semester (sections 4.2 to 4.5).

4.1 Submission clustering: typical behaviour on one submission
In order to determine the types of approaches students take when completing weekly tasks, we performed a clustering on all the data available. For each given student and week, we created a vector containing information about the student’s behaviour on that week’s submission. We chose features which related to student submission times as an indication of their approach to the task. We also included features relating to student marks, number of attempts and number of compile errors, which provided an indication of performance. In total there were 5434 vectors (11 weeks, 494 students), each representing a submission (possibly non-existent) by one student. Table 1 describes the features used in this initial clustering.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>percent_early</td>
<td>Percentage of attempts made three days or more</td>
</tr>
<tr>
<td></td>
<td>before the due date.</td>
</tr>
<tr>
<td>percent_normal</td>
<td>Percentage of attempts made that were neither</td>
</tr>
<tr>
<td></td>
<td>early nor late.</td>
</tr>
<tr>
<td>percent_late</td>
<td>Percentage of attempts made on the due date.</td>
</tr>
<tr>
<td>num_compile_errors</td>
<td>Number of attempts involving compilation errors.</td>
</tr>
<tr>
<td>first_mark</td>
<td>Percentage of tests passed on first attempt.</td>
</tr>
<tr>
<td>last_mark</td>
<td>Percentage of tests passed on last attempt.</td>
</tr>
<tr>
<td>num_attempts</td>
<td>Number of attempts not involving compilation</td>
</tr>
<tr>
<td></td>
<td>errors.</td>
</tr>
<tr>
<td>time_taken</td>
<td>Indicator for the time between the first and last</td>
</tr>
<tr>
<td></td>
<td>submission.</td>
</tr>
<tr>
<td></td>
<td>0: student only made 1 submission (time between</td>
</tr>
<tr>
<td></td>
<td>the first and last submission not relevant);</td>
</tr>
<tr>
<td></td>
<td>0.5: student took less than 26.45 minutes to</td>
</tr>
<tr>
<td></td>
<td>complete their task;</td>
</tr>
<tr>
<td></td>
<td>-100: student did not attempt the task; (forces</td>
</tr>
<tr>
<td></td>
<td>students who did not submit into their own</td>
</tr>
<tr>
<td></td>
<td>cluster)</td>
</tr>
<tr>
<td>single_attempt</td>
<td>Specifies whether the student made no attempts</td>
</tr>
<tr>
<td></td>
<td>(&quot;none&quot;), a single attempt (&quot;yes&quot; or multiple</td>
</tr>
<tr>
<td></td>
<td>attempts (&quot;no&quot;).</td>
</tr>
</tbody>
</table>
We note that the features, percent_early, percent_normal and percent_late are dependent. However, removing one would lead to different results depending on which feature was removed, so all were included to preserve symmetry.

We then clustered these 5434 vectors (with k-means algorithm) into six groups, with centroids are summarised in Table 2. Since these clusters would be used to perform further clustering, in which the distance between all clusters would be assumed to be equal, it was important that there were not two similar clusters, or one cluster comprised of what should be two clusters. We experimented with various numbers of clusters in the range of 4-7, and found that 6 clusters best satisfied these criteria.

Table 2. Cluster centroids of submissions clustering

<table>
<thead>
<tr>
<th>Feature</th>
<th>Full Data</th>
<th>Cluster Number (Number of Vectors)</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(5434)</td>
<td>(488) (1017) (903) (719) (607)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>% early</td>
<td>0.30</td>
<td>0.55 1.00 0 0.39 0.00 0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>% normal</td>
<td>0.22</td>
<td>0.19 0.00 0 0.43 0.99 0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>% late</td>
<td>0.17</td>
<td>0.27 0.00 0 0.18 0.01 0.99</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>num compile</td>
<td>0.14</td>
<td>0.79 0.08 0 0.06 0.17 0.21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>first mark</td>
<td>0.57</td>
<td>0.65 0.96 0 0.68 0.93 0.88</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>last mark</td>
<td>0.64</td>
<td>0.76 0.98 0 0.96 0.96 0.90</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>num attempts</td>
<td>0.44</td>
<td>0.59 0.52 0 0.94 0.54 0.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>time taken</td>
<td>-31*</td>
<td>0.78 0.07 -100* 0.74 0.17 0.18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>single attempt</td>
<td>yes</td>
<td>no  yes none no yes yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The features typical of each of the clusters allow us to interpret the general behaviour captured in these clusters. These are summarised in Table 3 and discussed in more detail below. Note that we refer to the following five grade categories from here on: High Distinction (HD), mark of 85 or above; Distinction (D), mark between 75 and 84; Credit (CR), mark between 65 and 74; Pass (P), mark between 50 and 64; Fail (F), mark below 50.

Table 3. Brief description of submissions clusters

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Typical Behaviour for the submission</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Early start, steady improvement from CR to D.</td>
</tr>
<tr>
<td>1</td>
<td>Early start, strong first attempt.</td>
</tr>
<tr>
<td>2</td>
<td>No submission made</td>
</tr>
<tr>
<td>3</td>
<td>Normal start, steady improvement from CR to HD.</td>
</tr>
<tr>
<td>4</td>
<td>Normal start, strong first attempt.</td>
</tr>
<tr>
<td>5</td>
<td>Late start, strong first attempt.</td>
</tr>
</tbody>
</table>

Cluster 0: Attempts in this cluster were started early and progressed for a long and had a high number of compile errors in the attempts. They contained a medium number of attempts, and their improvement was moderate: attempts began with around a credit and improved to a distinction. (9% of vectors were in this cluster).

Clusters 1, 4 and 5: these represent cases where students performed well in the weekly task and began early, neither early nor late, and late respectively. Students, when in any of these three clusters, on average began with an initial and final mark of HD. However, Cluster 1 students had the highest average mark in both cases (96-98), followed by Cluster 4 (92-96), then Cluster 5 (88-90). These students usually made a medium number of attempts with a small number of compilation errors over a small amount of time. (19, 13 and 11% of instances respectively).

Cluster 2: This cluster represents cases where students did not attempt the task. (31% of cases).

Cluster 3: The high number of submissions and time taken suggests students, when in this cluster, put in the most effort. Improvement was typically large – from around a low credit (68) to an HD (96). The majority of these students’ attempts were not late, and there were a low number of compilation errors. (17% of instances).

Intuitively, we would describe Clusters 0 and 3 as the behaviours that make best use of the autograding system, by making use of the feedback to achieve a significantly higher final grade.

Clusters 1, 4 and 5 are interesting because these behaviours are unlikely to benefit from being able to make multiple attempts, since early attempts are already of a high quality. It might be that students who found a task easy to complete in one week may not feel the need to invest time early in subsequent.

Figure 2 shows the general distribution of behaviours each week. We can see that many students were in Cluster 1 in the first week, probably due to the simplicity of the task, and that the number of students who did not submit at all (Cluster 2) is similar from week 2 to week 8, but increasing towards the end of the semester, especially in weeks 9, 10 and 11. This can be explained by the fact that these weeks are heavy in assignment deadlines in all the courses, including this course.
We chose to study the relationship of the submission clusters with the final exam since it is the main and most comprehensive assessment component in the course. It is worth 60% of the final mark, covers all topics and is highly correlated with the final mark for the course. Here we use the same grade categories as previously: HD, D, CR, P, F. NA denotes students who did not sit the exam. There is a minimum requirement policy of scoring at least 40% in the final exam to pass the course: this means that even if students scored very high during the semester (say, 100% of 40), they would fail the course if they scored less than 40% at the final exam (say 30% of 60), even though their raw mark would be above a pass (58%).

We can see that the students who obtained HD and D in the exam were often in Cluster 1 during the semester and also sometimes in Clusters 4 and 3. These clusters corresponding to the best performing students during the semester, with Cluster 1 containing the students who start early with a very high initial mark, Cluster 4 – the students who start normally with a high mark and Cluster 3 – the students who start early or normally from an average mark and work very hard to improve their submissions.

The students who obtained CR and P at the exam did not show a predominant behavioural pattern during the semester when completing the weekly tasks – they belonged to all clusters. However, more P than CR students were in Cluster 2 (the cluster of students who did not submit), for all weeks. In contrast, very few of the CR students were in Cluster 2 in the early weeks although this number increased after week 8.

A large proportion of the students who failed the exam were in Cluster 2 during the semester, but there are failing students in all behavioural clusters. The students who did not sit the exam are predominantly from Cluster 2 and, from Figure 1, their number is relatively stable from week 2 to week 12, which shows that most likely these students dropped out early in the semester.

### 4.3 Evolution of students from a given cluster

We can also follow the evolution of the students from a given cluster from a specific week. For example, starting with the six clusters from Week 3, we can analyse each cluster separately and investigate where the students from each cluster go in the subsequent weeks, as shown in Figure 2.

![Figure 3. Percentage of submissions in each submission cluster each week with the submitting student’s final exam grade](image-url)

The graphs show that the students from Cluster 0 in week 3 were mainly in Clusters 1 and 3 in the following weeks, i.e. they were able to achieve a higher mark on the weekly tasks compared to week 3. The students from Cluster 1 in week 3 mainly stayed in the same cluster or moved to Cluster 3, i.e. had to put more effort to maintain high marks. The students from Cluster 2 in week 3 (the non-submitting students) stayed in the same cluster with very few exceptions. The students from Clusters 3 and 4 together stayed in these clusters, and the students from Cluster 5 in week 3 moved...
between Clusters 3, 5 and 2 during the semester, i.e. they were not always able to achieve high mark, possible because they started late, and also did not submit in some weeks, e.g. week 10.

We can clearly say that extracting patterns based on visual analysis of the graphs is difficult. This motivated our second clustering of behavioural data described in the next section.

### 4.4 Comparing the clusters in the middle and end of the semester

To better understand the stability of the clusters over time, we conducted clustering in the middle of the semester (after week 7) using the same method as described in Sec 4.1. We then compared the new clustering to the old clustering, described in Sec 4.1, to determine whether the end-of-semester clusters had already formed midway through the semester. Note that the clustering in both cases is done using all the available data at that time point, i.e. the mid-semester (early) clustering uses the data from week 2 to week 7, and the end-of-semester (end) clustering uses the data from week 2 to week 12.

In both cases, we followed the same clustering procedure – one example represents one submission. We paired each early cluster with a corresponding end cluster, seeking to maximize the overlap between the matched clusters.

More precisely, we considered the bijection, $m$, from the set of end clusters to the set of early clusters which minimized the distances between the centroids of each late cluster $c_i$ and the paired early cluster $m(c_i)$. We then defined the accuracy of $m$ on an early cluster $m(c_i)$ to be the proportion of submissions in end cluster $c_i$ that were also in early cluster $m(c_i)$. That is,

$$\text{accuracy}(m(c_i)) = \frac{|S(m(c_i)) \cap S(c_i)|}{|S(c_i)|}$$

where $i$ is an integer from 0 to 5, $S(x)$ denotes the set of submissions assigned to the cluster $x$, and $|X|$ denotes the number of elements in set $X$.

The chosen bijection gives the accuracies shown in Table 4. We can see that the accuracy of the mapping of four of the end clusters (1, 2, 3 and 5) is very high ($\geq 90\%$). This is to be expected of Cluster 2 as all non-attempts are forced into their own cluster. However, this is not the case for Cluster 1, Cluster 3 and Cluster 5, and the high accuracy indicates that these clusters had already formed midway through the semester. End Cluster 4 had also emerged in week 7, as evident by relatively high accuracy of the mapping to it (76%), but had not stabilized yet. The mapping of end Cluster 0 had a low accuracy, indicating that this cluster had not yet been formed in week 7. A closer examination shows that the students in early Cluster 0 used strategies typical not only of end Cluster 0 but also of end Clusters 1 and 4, as well as end Clusters 5 and 3, to a lesser extent.

### Table 4. Accuracy of each cluster in the middle of the semester (week 7) relative to the end of the semester (week 12)

<table>
<thead>
<tr>
<th>End cluster (week 12)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy in week 7</td>
<td>13%</td>
<td>90%</td>
<td>100%</td>
<td>91%</td>
<td>76%</td>
<td>97%</td>
</tr>
</tbody>
</table>

Before we describe these clusters, we also examined the relationship between final exam marks and a student’s behavioural cluster. Figure 3 shows the percentage of students in each behavioural cluster receiving each of the possible exam grades: HD, D, CR, P, F and NA, where NA indicates that a student did not sit the final exam. The behavioural clusters in this figure have been ordered from lowest to highest based on the percentage of students passing the final exam in those clusters (i.e. behavioural clusters 3, 4, 1, 5, 2, then 0). We see in general that the proportion of passing students that receive higher bands increases, as well as the proportion of students who sit the final exam.

### 4.5 Behavioural evolution in time

The submission clustering in section 4.1 gave us clusters capturing behaviour per student per weekly task. An interesting question is how each student’s behaviour evolved during the semester in regards to their weekly task. In order to explore this question, we performed an additional clustering to identify groups of students with similar submission behaviours over the weeks. The features used for this clustering try and capture the variety and frequency of behaviours (in terms of submission clusters found in 4.1). Note that features, c0-c5 count, are dependent, since the number of weeks are fixed. However, as previously, we maintain all to preserve symmetry. These features are described in Table 5. K-means clustered students into 6 groups, where the number of clusters was determined empirically. The centroids of this new clustering, which we call behavioural clustering, are shown in Table 6.

### Table 5. Features used in behavioural clustering

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>num_clusters</td>
<td>Number of submission clusters a student’s submission belonged to over the semester</td>
</tr>
<tr>
<td>c0_count</td>
<td>Number of weeks where a student’s submission belonged to behavioural cluster 0</td>
</tr>
<tr>
<td>c1_count</td>
<td>Number of weeks where a student’s submission belonged to behavioural cluster 1</td>
</tr>
<tr>
<td>c2_count</td>
<td>Number of weeks where a student’s submission belonged to behavioural cluster 2</td>
</tr>
<tr>
<td>c3_count</td>
<td>Number of weeks where a student’s submission belonged to behavioural cluster 3</td>
</tr>
<tr>
<td>c4_count</td>
<td>Number of weeks where a student’s submission belonged to behavioural cluster 4</td>
</tr>
<tr>
<td>c5_count</td>
<td>Number of weeks where a student’s submission belonged to behavioural cluster 5</td>
</tr>
</tbody>
</table>

### Table 6. Behavioural cluster centroids

<table>
<thead>
<tr>
<th>Feature</th>
<th>Full Data</th>
<th>Behavioural Cluster Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>num_clusters</td>
<td>3.92</td>
<td>0</td>
</tr>
<tr>
<td>s0_count</td>
<td>0.99</td>
<td>1.00</td>
</tr>
<tr>
<td>s1_count</td>
<td>2.06</td>
<td>5.26</td>
</tr>
<tr>
<td>s2_count</td>
<td>3.44</td>
<td>0.69</td>
</tr>
<tr>
<td>s3_count</td>
<td>1.83</td>
<td>2.22</td>
</tr>
<tr>
<td>s4_count</td>
<td>1.46</td>
<td>1.42</td>
</tr>
<tr>
<td>s5_count</td>
<td>1.23</td>
<td>0.42</td>
</tr>
</tbody>
</table>
We note that over 80% of students in Behavioural Cluster 0, which comprised 20.4% of the cohort, passed the final exam – the highest percentage of all the secondary clusters. In addition, over 50% of students in this behavioural cluster received at least a credit.

Behavioural Cluster 2 had the next highest pass rate of around 70%. The proportion of students receiving high bands in this cluster was lower than Behavioural Cluster 1, but greater than in other clusters.

Using the cluster centroids in Table 6, the weekly behaviours typical of different behavioural clusters are summarised below, in the cluster order used in Figure 3.

Behavioural Cluster 3: These students belonged to an average of 1.3 different clusters throughout the semester. 96.8% of the time they were assigned to Submission Cluster 2, indicating that they almost never completed their weekly tasks. These students may have dropped out of the course during the semester. 16.2% of students.

Behavioural Cluster 4: These students oscillated between an average of 3.5 clusters throughout the semester. 65.4% of the time, they fell into submission Cluster 2, indicating that they frequently did not complete their weekly tasks. However, these students belonged to submission Cluster 5 11.6% of the time, suggesting they sometimes started late but still performed well. From this, we see that these students are possibly quite capable, but do not put much effort into their weekly tasks.

Behavioural Cluster 1: These students were in an average of 5.1 submission clusters over the semester. Cluster 5 was the most common submission cluster, which students were in 26.3% of the time, followed by Cluster 2 (19.3%), Cluster 3 (14.6%), Cluster 0 (13.8%), Cluster 1 (13.1%) and Cluster 4 (13.0%). Thus these students often started late but did well, but also often didn’t submit at all. These students sometimes worked hard and achieved high marks, sometimes worked hard without achieving high marks, sometimes began early and did very well and sometimes began neither early nor late and did well. These students displayed inconsistent behaviour over the weeks, sometimes putting in a great amount of effort and sometimes not trying at all. 24% of students.

Behavioural Cluster 5: These students belonged to an average of 4.4 different clusters over the semester. They fell into submission Cluster 4 the most often - around 41.1% of the time – followed by submission Cluster 3 (19.5%), Cluster 1 (13.5%) and Cluster 5 (12.8%). Thus these students very often started their weekly tasks neither early nor late and did well, commonly started early and worked hard until they did well, sometimes started early from a high mark and sometimes started late from a high mark. (13% of students)

Behavioural Cluster 2: These students belonged to an average of 4.4 different submission clusters over the semester, with Cluster 3 being the most common (40.8%), then Cluster 1 (20.1%), Cluster 0 (13.6%) and Cluster 4 (11.9%). Thus, these students commonly began early with a medium mark, worked hard and achieved good marks. They also often started early from a high mark, sometimes worked hard without achieving a high mark and sometimes started neither late nor early with a high mark. These are hard-working students who often found the tasks challenging, but still did fairly well in them.

Behavioural Cluster 0: Finally, in the behavioural cluster with the highest final exam pass rate, students oscillated between an average of around 4.2 clusters in the course of the semester. They were in submission Cluster 1 47.8% of the time, Cluster 3 20.2% of the time, Cluster 4 12.9% of the time and cluster 0 9.1% of the time. This suggests these students started early with high marks around half the time. They often started early with medium marks, but worked hard until they achieved a high mark and sometimes started neither late nor early, achieving high marks. Occasionally they worked hard without achieving high marks. (20% of students). These students often did well on their first submission but, when they didn’t, they worked hard to achieve high marks.
and so came next on the scale. This was followed by Submission Clusters 4, 3 and then 1, which became more prevalent in higher performing behavioural clusters. Figure 4 shows the percentage of submissions in each behavioural cluster that fell into each submission cluster. The behavioural clusters are ordered based on pass rate, and the submission clusters are ordered as described above. The prevalence of each submission cluster in different behavioural clusters is summarised in Table 7.

Table 7. Submission Clusters Typical of each Behavioural Cluster

<table>
<thead>
<tr>
<th>Submission Cluster</th>
<th>Common in Behavioural Clusters with</th>
<th>Submission Cluster Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Many different pass rates</td>
<td>Average students, medium/high effort.</td>
</tr>
<tr>
<td>1</td>
<td>High pass rates</td>
<td>Excellent students who started early from a very high mark.</td>
</tr>
<tr>
<td>2</td>
<td>Low pass rates</td>
<td>Did not submit.</td>
</tr>
<tr>
<td>3</td>
<td>High pass rates</td>
<td>Hard working students – from CR to HD.</td>
</tr>
<tr>
<td>4</td>
<td>Medium pass rates</td>
<td>Good students who started neither early nor late from a mid HD.</td>
</tr>
<tr>
<td>5</td>
<td>Low pass rates</td>
<td>Good students who started late from a low HD and improved slightly.</td>
</tr>
</tbody>
</table>

4.5.2 The median
We can also visualise the evolution of student behaviour over the semester in a meaningful way. We looked at the weekly behaviour of students in each behavioural cluster each week and found the “median” behaviour. This was achieved by taking the median of each original feature for these students, such as the first mark, last mark, time taken and percentage of early submissions. We then used this to create a median vector, and found which submission cluster the vector belonged to. We repeated this for all behavioural clusters and plotted the results. This can be seen in Figure 5. Note that submission clusters were previously ordered so the higher the submission cluster the more typical it is overall of the behavioural clusters with the highest pass rate.

Figure 5. Changing student behaviour over the semester. Each colour represents a behavioural cluster. The median behaviour of students each week (i.e. the median submission cluster) is shown. The submission clusters are ordered so that higher corresponds to better performance.

Rather than the secondary clusters slowly diverging over time, we notice a clear separation from as early as week 3. The secondary clusters with the lowest (secondary clusters 3 and 4) and highest (secondary cluster 0) pass rates are already distinguishable from the other clusters at this time. This early separation of behaviours could facilitate early identification of students at risk of failing or performing poorly, allowing for intervention.

5. DISCUSSION
The scheme in our analysis can be separated into two parts:
(i) A submission clustering, where the approach and performance of each student in each weekly submission is treated as independent and then clustered to give typical task-level behaviours.

(ii) A behavioural clustering, where students are clustered based on the submission clusters they were in over the entire semester.

Through the example of a junior computer science course, we demonstrated the usefulness of this double-clustering method in allowing us to identify some important approaches students in this course took to their weekly tasks. We found that many students started sufficiently early and invested time to improve their attempts based upon instant feedback they received from the autograding system, benefiting from a significant improvement in the quality of their final attempts (Clusters 0 and 3, 26%). We also found that students often found the task sufficiently easy and that further improvements were of little value (Clusters 1, 4 and 5, totaling 43%), and that it was also common for students to not attempt the tasks at all (Cluster 2, 31%). A broader application of this analysis over multiple units of study and across multiple offerings of the same course would be useful in understanding how common such behaviours are in general as opposed to this specific offering.

Through the behavioural clustering, we were able to identify common behavioural patterns over the entire semester, and to draw links between these patterns and final exam outcomes. In particular, we identified behavioural patterns associated with high and low final exam grades. For example, students in behavioural clusters with high pass rates tended to consistently start early with a high mark, or start early and work hard until a high mark was achieved. Conversely, students in behavioural clusters with low pass rates often did not submit their tasks at all. Knowledge of the relationship between behavioural patterns and exam performance is essential in the identification of students at risk of performing poorly and important in the structuring of a course to maximise student learning and performance.

We compared submission clusters that used all data up to week 12 with submission clusters that used all data up to week 7, and found that they were quite similar. This suggests that the typical task-level behaviours of students did not vary much at the end of the semester and that, as a consequence, these behaviours could be identified early on in the semester. Moreover, we saw that the term-long behavioural clusters we found did not slowly diverge over time, but rather there was an immediate difference from as early as week 3. This suggests that both the submission and behavioural clustering could be performed early in the semester, with potentially similar results to the end of semester, allowing for early identification of students at risk of performing poorly and early intervention. We suggest an avenue of future research could be to apply this technique midway through the semester and evaluate its
effectiveness in facilitating interventions that could improve student outcomes.

We also suggest investigating how effective this method can be in general, by applying it to courses with different assessment structures and content, and also to compare the results obtained through these clustering methods to traditional measures of behaviour and engagement, such as tutorial attendance and feedback surveys, to evaluate how well they corroborate.

Although the reported analysis is for data from a system for assessing computer code submissions, it could just as readily be applied to other systems in which students can make multiple submissions in response to feedback. For instance, many Learning Management Systems provide multiple-choice style questions for which students can receive feedback about their choices, and this style of question could be used in any discipline. Our analysis depends only upon records of the time and quality of each submission. While we include details such as number of compile errors as one measure of quality, this could readily be substituted with other measures.

6. CONCLUSION
In this paper we have presented a method for analysing student behaviour and the evolution of this behaviour over the semester, using data from autograding system logs. We have shown that this method can be useful in identifying common weekly behaviours of students, and following the changes of such behaviours over the semester. We have discussed the relationship between these behaviours and final exam results, and demonstrated how these behaviours might be detectable early enough in the semester for instructors to intervene. As such, we believe that the techniques discussed here may be implemented and improved upon to realise the full potential of increasingly common autograding systems in facilitating real improvement in student outcomes.

7. ACKNOWLEDGMENTS
This work was funded by the Human-Centred Technology Cluster of the University of Sydney.

8. REFERENCES
Modelling the way: Using action sequence archetypes to differentiate learning pathways from learning outcomes

Kelvin H R Ng  
Nanyang Technological University  
50 Nanyang Ave  
Singapore 639798  
e140025@e.ntu.edu.sg

Kevin Hartman  
Nanyang Technological University  
50 Nanyang Ave  
Singapore 639798  
khartman@ntu.edu.sg

Kai Liu  
Nanyang Technological University  
50 Nanyang Ave  
Singapore 639798  
kliu006@e.ntu.edu.sg

Andy W H Khong  
Nanyang Technological University  
50 Nanyang Ave  
Singapore 639798  
andykhong@ntu.edu.sg

ABSTRACT
During the semester break, 36 second-grade students accessed a set of resources and completed a series of online math activities focused on the application of the model method for arithmetic in two contexts 1) addition/subtraction and 2) multiplication/division. The learning environment first modeled and then supported the use of a scripted series of steps for solving mathematical word problems. As students completed the activities, the learning environment captured their event-related data. We then used a combination of Affinity Propagation, an automated form of clustering, and sequential pattern mining to convert the activity logs into interpretable activity sequences. Analysis of the activity sequences identified distinct patterns of behavior that strongly predicted which students would transit from the familiar addition/subtraction word problem activity to the unfamiliar multiplication/division word problem activity. Students who showed the greatest and least compliance with the script were the least likely to attempt the multiplication/division activity. Students who showed more of a schematic problem solving process were more likely to continue to the multiplication/division activity.

Keywords
Sequential pattern mining, affinity propagation, cognitive models

1. INTRODUCTION
1.1 Mathematics Learning via the Model Method
In Singapore, early-elementary students are taught to solve arithmetic word problem via the model method [1]. This systematic approach is based on Polya’s problem solving techniques [2]. The method can be broken into five steps known as the RIGHT sequence. When applying the RIGHT sequence, students 1) read the word problem, 2) identify the nouns, numeric values, and unknown variable to be solved, 3) graph these values in a box diagram, 4) indirectly perform the appropriate calculation by reasoning through the diagram, and 5) review their work.

The RIGHT sequence, as a learning mnemonic, provides students with a script for executing the model method. Scripts are collections of discrete actions that, when followed, achieve a goal or specific outcome [3]. Ordering food at a restaurant serves as the classic example of following a cognitive script [3]. In most dining establishments, the same set of steps, with some allowance for minor deviations, will lead the patron to receive a meal. Similarly, following the RIGHT sequence will lead students to the correct answer to a word problem. Scripts have been found to reduce cognitive load for novice learners by lessening the mental resources needed for planning and completing the plan. Scripts also lead to greater expressions of automaticity by experts [4]. However, cognitive psychologists also view scripts as the most nascent form of schemas [5]. The application of scripts is contextually bound and rather inflexible. Schank and Abelson [6] refers to scripts as event schemas which are task specific and order dependent. The previous restaurant script may work for purchasing food at most dining establishments, but it could not be used successfully to purchase food at a supermarket. To negotiate the supermarket, one would need to apply either a different script or rely on a more generalizable schema.

Generalizable schemas consolidate the steps of an event schema under a larger label [7]. Rather than simply ordering a meal at a restaurant, a generalizable schema for acquiring food would include all of the known methods of gaining nourishment. What generalizable schemas sacrifice in terms of automaticity, they make up with flexibility [5].

Returning to the original example of the model method, the intent behind introducing students to using box diagrams to solve algebraic word problems is to give them a generalizable schema for solving real-world problems [1]. In practice, students often instantiate the schema in the form of a word problem solving script [8]. When looking at problem solving accuracy, teachers cannot diagnose whether a student has internalized the model method as a generalizable schema or as a problem solving script because both strategies work in the short term. However, only the generalizable schema prepares students to flexibly transfer the model approach to new situations. In this study, we sought to generate an algorithm to classify students as exhibiting script-like or generalizable schema-like behaviors in the context of a series of online math enrichment activities. We then tested whether script-like behaviors, generalizable schema-like behaviors, or problem solving accuracies were more predictive of students seizing future learning opportunities.

1.2 Machine Learning and Temporal Sequencing
In the context of this paper, we define an action as a single line item in a log file and action sequences as the collection of actions that can be described with a more general semantic label. For instance, entering a number into a text box constitutes an action. All of the various combinations of actions that lead to the calculation of that
When attempting to identify meaningful action sequences while preserving the temporal relationships between those actions, educational data miners use techniques like process mining and sequential pattern mining. With process mining, the learning pathways students take within a learning environment are identified and visualized [9, 10]. Deviations in these pathways from the intended pathways can then be analyzed for meaning [9, 10]. Alternatively, sequential pattern mining identifies frequently occurring subsequences within a temporal dataset for further analysis. Recently, Ye et al. used a hierarchical variant of SPAM to analyze data collected from Betty’s Brain OLE [11]. The analysis illustrates the importance of using temporal relationships between user activities to make predictions about future learning behaviors [11]. Veeramachaneni, Adl, and O’Reilly [12, 13] also highlight the significance of incorporating a range of temporal dependencies into features when predicting student traits. Applying a crowd sourcing technique, they obtained lists of complex features that, when divided, seem obvious to experienced teachers and data scientists. However, neither group could have generated the entire list of the features on its own [12].

When extracting frequent patterns from unstructured data, sometimes the patterns are composed of short sets of actions which actually belong to longer action sequences. These algorithms have a tendency to obscure the temporal relationships between the extracted features. Additionally, sequencing combinations of actions and filtering out rare patterns rather than using the complete action sequences can result in the loss of rare action combinations that achieve a common action sequence [14]. The potential for losing rare actions belonging to common action sequences is magnified in learning environments populated by novice learners. Novice learners who are introduced to a learning environment have the dual task of learning to navigate the environment as well as gaining competency with the concepts central to the learning activities. In such situations, data mining techniques that analyze learner actions more schematically, rather than in scripted terms, may actually yield more parsimonious models.

With the goal of aligning our data mining techniques with learners’ mental schemas, we propose conceptually reframing individual actions as words and action sequences as sentences. With this recasting, we can apply a combination of string distance measures that take into account the vocabulary and word order within the sentences to make pair-wise comparisons. We used an Affinity Propagation (AP) [15] algorithm to recover distinct action sequences that translated to learning behaviors and the sequence exemplars are referred to as action sequences archetypes (ASAs). Sequential pattern mining is applied to cluster members to summarize the temporal deviations within each cluster. The described method preprocesses the data for analysis and interventions to steer learners towards desired educational outcomes.

AP is useful for our particular context because it simultaneously considers all data points in relation to a shared preference to determine a suitable number of output clusters. This structure independence lends AP to situations where there is no a priori expectation about the output cluster size or number [15]. In our case, the number of sequences within the dataset varies greatly between sessions. Beyond accommodating this variability, the algorithm’s input, a similarity matrix defined by the pairwise similarities between two sequences, is not limited to symmetrical pairwise similarities. This freedom creates opportunities to differentiate the discrete ordered lists using different distance measurements. We augmented the AP algorithm with a tree-based sequential pattern mining algorithm for its ability to handle multiple minimum supports and rare item filtering [14]. The algorithm is used to extract maximal sequences, which are longest sequences that satisfy the minimum frequency threshold, for each cluster.

2. Data Collection

36 second grade students completed the first phase of activities in the online learning environment during the school holidays. The activities were part of an “out of school” enrichment opportunity. At the onset of data collection, all of the invited participants had previously received formal instruction from their teachers on using the model method to solve addition and subtraction word problems. The students had not yet received instruction within the school curriculum on using the model method with multiplication and division word problems.

The online learning environment offers two phases of content. During Phase 1, students’ complete addition and subtraction activities. In Phase 2, students encounter multiplication and division activities. Each content phase is divided into four sets of activities: 1) video tutorials, 2) structured activities, 3) unstructured activities, and 4) multiple choice questions (MCQ). The video tutorials explain the RIGHT sequence and the use of the model method in a pen-and-paper context. After each video, students receive a set of practice exercises related to the content of the video tutorial. Additional video segments at the start of each practice question introduce the recommended sequence of steps to solve the word problems using the model method and the representational supports found within the learning environment. The representational supports include using the highlighted noun blocks and the RIGHT checklist while answering the word problems.

The structured activity focuses on the “G” in the RIGHT sequence. Each question in the activity is presented with a practice word problem. The problem is displayed with four multiple choice options showing different bar diagrams and a checklist in the right corner of the workspace. The checklist shows the first three steps of the RIGHT sequence. Students are advised to tick off the respective check boxes as they complete each step in the RIGHT sequence. In the structured activity, the checklist is limited to the first three steps of the RIGHT sequence as students are not expected to take their model to completion.

After students identify the model they think matches the content of the word problem, they are given feedback about their choice before moving on to the next question. They are presented with options to review, ask for hints or proceed to the next question. Choosing to review the question returns students to the last snapshot of the question before the answer submission. Requesting a hint provides students with a partially completed model as a guide. Hints are given progressively until the complete model is revealed. Two hints can be requested for each question. If a student chooses to proceed to the next question without reviewing errors after submitting an error, the learning environment logs the action as ignoring an error.

In the unstructured activity, students solve the problems using the RIGHT sequence. A snapshot of the learning environment for this activity prior to any attempt is shown in Figure 1. Model templates for all four arithmetic operations are made available for students to complete with the correct numerical values. Nouns mentioned in the problem are also presented as colored blocks for labeling the relevant model. Students can drag and drop the blocks to their
selected model. Students may also enter mathematical expressions in the provided text boxes. Alternatively, students may forego performing any or all of these actions. However, they must submit a final answer before receiving feedback about their answer and proceeding to the next question.

For the MCQs, students are presented with a page containing ten multiple-choice questions. Each question requires inputting a numerical answer into a textbox. Students again have the option of using the RIGHT checklist that floats in the right margin of the screen. The checklist resets whenever a student interacts with a different question. Students must complete all of the activities before proceeding to Phase 2.

3. Data Preprocessing

Only clickstream and navigation information occurring within the online module was recorded to the log file as students worked through the activities. Beyond navigation and interface information like mouse clicks and text entries, off-task behavior like leaving the learning environment by activating another browser tab and returning to the online module was also collected. A total of 23233 log entries were collected. Table 1 lists the recorded actions.

The log entries were preprocessed to indicate the use of the different learning resources within the learning environment. For example, highlighting a keyword within a question is recorded as one log entry per keyword. However, only the first instances of highlighting and canceling of highlights are retained for each question attempt to signal that the highlighting resource was used. In addition, while learners navigate through the model template selection, we only analyze the final template selection instead of considering all of the navigation activity within the selection area. Filtering out these events greatly reduces the amount of variability within the action sequences and makes them more schematic. To identify revision of answers, first selections for the MCQs are labelled as mcq_select. Additional selections are labelled as mcq_alter. Following the described procedure reduced the size of the dataset to 9918 entries, or 275 entries per student. The maximum number of analyzed actions for a student was 868. The final list of actions for each type of activity is shown in Table 1.

In the reduced dataset, each action sequence is identified and labelled. For videos, an action sequence constitutes the actions taken from the start of a video to terminating the video either by completing the video or navigating away from the current page. For the exercises, the action sequences span from the initiation of a question until the user proceeds to the next question.

Table 1: List of all log actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Video</th>
<th>Structured</th>
<th>Unstructured</th>
<th>MCQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>leave_page</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>return_to_page</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>phase_start</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>phase_stop</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>video_start</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>video_stop</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>video_pause</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>video_scrub_forward</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>video_scrub_back</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>video_end</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>video_end_full</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>video_replay</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>video_select_same</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>video_select_diff</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>attempt_qn</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>highlight</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>undo_highlight</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>check_checklist</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>mcq_select</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>mcq_alter</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>confirm_model</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>mouse_drag</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>label_model</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>label_eq</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>submit</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>review_error</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ignore_error</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>show_hint</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

1 Actions are inferred from clickstream data due to limitation of YouTube’s application programming interface (API).

2 Actions are recorded but filtered out for the purpose of this analysis.

4. Techniques

4.1 Distance Measures

To differentiate action sequences as one would differentiate sentences, it is necessary to consider the vocabulary (actions) of each action sequence and the order of those words. Our proposed distance measure includes four components, a modified version of the common word order measure [16], Jaccard distance, length difference, and vocabulary rarity. The features capture different aspects of action sequences for differentiation. The distance measure between two action sequences $S_1$ and $S_2$ is given by the weighted sum of all four features. In this paper, a constant weight is assigned across the four features.

$$
\text{dist}(S_1, S_2) = w_1 \cdot \text{JaccardDist}(S_1, S_2) + w_2 \cdot CWO(S_1, S_2) + w_3 \cdot \max \left( d_{f_1,s_1}, n_{s_2}(t_j, D) \right) + w_4 \cdot \abs{\text{length}(S_1) - \text{length}(S_2)}
$$

(1)

where

$$
w_1 = w_2 = w_3 = w_4 = 1
$$

(2)
Jaccard distance defined by

\[ JaccardDist(S_1, S_2) = 1 - JaccardSim(S_1, S_2) \]  (3)

where

\[ JaccardSim(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} \]  (4)

captures the degree of dissimilarity between two sequences through the number of unique terms that are not common to both. The Jaccard distances are derived from Jaccard similarity which determines the ratio of unique common actions between two action sequences. Jaccard similarity and distances are bounded between zero and one.

In our context, the common word order measure reflects the similarity of the order in which actions appear between two action sequences. The measure equals zero when the common actions of two sequences occur in the same order and reaches a maximum of one when the common actions appear in reverse order. Given two sequences \( A = \{a_1, a_2, ..., a_n\} \) and \( B = \{b_1, b_2, ..., b_m\} \) composed of \( l \) common action, where \( l \leq n \leq m \). Retaining only the common actions, sentence \( A = \{a_1, a_2, ..., a_l\} \) is transformed into a numerical representation \( X = \{1, 2, ..., l\} \) by substituting the actions with its indices. The same actions in \( B \) are replaced with the same numerical indices to form \( B \). The common word order measure can then be computed by

\[
CWO(S_1, S_2) = \begin{cases} 
1 - \frac{2 \sum_{i=1}^{l} |x_i - y_i|}{l^2}, & \text{if } l \text{ is even} \\
1 - \frac{2 \sum_{i=1}^{l} |x_i - y_i|}{l^2 - 1}, & \text{if } l \text{ is odd} \\
1, & \text{if } l \text{ is odd and } l = 1
\end{cases}
\]  (5)

The common word order measure is designed for sentences where a bag-of-words representation has a large number of words, most of which have low frequencies. Due to the constraints of the learning environment, our data set contained many actions with high frequencies. Retaining the common terms within action sequences may result in substrings of inequivalent lengths. Therefore, there may exist more than one combination of mapping between these sentences. To remedy this possibility, we adapted the concept of a common word order measure to obtain a distance estimate for the action sequences by first filtering the reduced sequences to remove actions occurring at a specific position that do not contribute to the distance metric. We then match the remaining actions based on their position within the reduced sequence.

The vocabulary rarity is defined as the maximum of the inverse document frequency (idf) \([17]\) of terms that are not common to both sentences. This measure allows us to distinguish sequences that have actions that are less likely to occur from sequences involving trivial navigational patterns. The inverse document frequency of each term \( t_i \) in a set of documents \( D \) is computed by the logarithmic inverse of the ratio of document counts containing \( t_i \) to the total number of documents in the document set \( D \).

\[
idf(t_i, D) = \log \frac{|D|}{|\{d \in D : t_i \in d\}|}
\]  (6)

4.2 Affinity Propagation

The AP algorithm \([15]\) is a message passing clustering algorithm used in image recognition, text comparison and gene clustering. Unlike centroid-based clustering like k-means clustering, AP does not require users to pre-specify the number of clusters and it is less sensitive to parameter initialization \([15]\). The algorithm takes a pair-wise similarity matrix and a set of shared preferences as inputs to determine the suitability of data points as cluster centroids. Without prior knowledge of the centroids, shared preferences may be set uniformly across all items. When shared preferences are assigned to the minimum value of the pairwise similarity, the number of resulting clusters will also be at its lowest. The inverse is also true. The number of clusters generated by the different shared preference values for the structured activity are shown in Figure 2.

![Figure 2: Number of generated clusters based on shared preferences for structured activity.](image)

For our purposes, clusters are determined by passing messages between data points (action sequences) to simultaneously determine their suitability as cluster centroids. The provided similarity matrix may contain unknown pair-wise similarities. However, messages are passed only between points with known similarities. There are two types of messages passed between data points -- responsibility and availability. Responsibility \( r(i, k) \), sent from data point \( i \) to data point \( k \), dictates the amount of evidence that \( k \) is suitable to serve as the exemplar for \( i \), while availability \( a(i, k) \), sent from \( k \) to \( i \), determines the appropriateness for point \( i \) to choose point \( k \) as its exemplar.availabilities are initialized as zeroes and the messages are updated iteratively using

\[
r(i, k) = s(i, k) - \max_{k' \neq k}(a(i, k') + s(i, k')), \quad (7)
\]

\[
a(i, k) = \min \left\{ 0, r(k, k) + \sum_{i' \in [k]} \max\{0, r(i', k)\} \right\} \quad (8)
\]

\[
a(k, i) = \sum_{i' \in [k]} \max\{0, r(i', k)\} \quad (9)
\]

At the end of each iteration, exemplars are determined from

\[
exemplar(i, k) = \arg\max_k(a(i, k) + r(i, k)) \quad (10)
\]

Pairs \((i, k)\) identified from equation (10) state that either data point \( i \) will serve as an exemplar for data point \( k \) or vice versa. The algorithm terminates only when either a predefined number of iterations is completed or the changes in the messages falls below a certain threshold.

Essentially, the AP algorithm seeks to identify action sequence archetypes (ASA) around which to cluster the remaining action...
sequences. After identifying the ASAs, the similar cluster sequences inherit the index of their closest archetype.

**4.3 Sequential Pattern Mining**

The position coded pre-order linked web access pattern tree mining (PLWAP) algorithm with multiple minimum supports (MMS) [14] is a tree-based sequential pattern mining algorithm. A PLMS-tree is constructed from the logs by adding actions for each learning opportunity sequentially. Each node holds four variables, the label, the frequency count, a binary position code, and a minimum multiple item support (minMIS).

The binary code is similar to Huffman coding as it uniquely identifies nodes and subtrees. The root node of the tree is labelled as 0. The leftmost child of any node has a position code of 1 appended to the back of the position code of the node. The position codes of other children are derived from the position codes of their nearest sibling to the left by appending a 0 to the position code.

The support determines the lower bound for frequencies that sequences must satisfy to qualify as a frequent pattern. For multiple minimum support, a minimum support is computed for each unique item in the dataset. In the case of our action sequences, the items in sequential pattern mining correspond to actions in the action sequences. The global minimum support is dictated by the smallest of the minimum supports. Each node maintains its minMIS which defines the support required by itself and the suffix tree to qualify as frequent.

As the nodes are added to the tree, a header table is maintained. The header table contains the unique node labels with a list of corresponding binary code of nodes for the same label within the tree. The table is then sorted by order of decreasing frequencies. An example of the PLMS-tree and its corresponding header table is shown in Table 2 and Figure 3.

**Table 2: Header table example for Figure 3**

<table>
<thead>
<tr>
<th>Label</th>
<th>Support</th>
<th>Position Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>video_start</td>
<td>10</td>
<td>{01}</td>
</tr>
<tr>
<td>video_end_full</td>
<td>6</td>
<td>{011}, {0110001}</td>
</tr>
<tr>
<td>video_end</td>
<td>4</td>
<td>{0110}, {011001}, {011000101}</td>
</tr>
<tr>
<td>video_pause</td>
<td>2</td>
<td>{01100010}</td>
</tr>
<tr>
<td>video_scrub_back</td>
<td>1</td>
<td>{01100010}</td>
</tr>
<tr>
<td>video_scrub_forward</td>
<td>1</td>
<td>{01100}</td>
</tr>
</tbody>
</table>

Once the tree is populated, it can be traversed to mine the sequential patterns in the dataset. The mining algorithm proceeds as follows:

1. For each of the entries in the header table, the nodes are identified from the tree using the position codes and the total occurrences is consolidated from the counts of individual nodes. A k-sequence is an ordered list of k items.
   a. If this sequence satisfies its minMIS, it qualifies as a 1-sequence.
   b. If the frequency of this node satisfies the global minimum but not its minMIS, the label qualifies as a 1-sequence candidate. Candidates are kept as candidates for mining because a subsequent item of lower minMIS may qualify these sequences as frequent sequences.

2. The algorithm proceeds to identify the next item in the sequence by scanning the header table.

3. Position codes in the header table containing the position code of the last found node as its prefixes are identified as descendants for that node.

4. The frequencies of the newly identified nodes are aggregated and a new minMIS is updated to be the lower of the minMIS from previous nodes and the identified node.

5. The algorithm proceeds to search for possible extensions and validates the frequency of these sequences against the minMIS.

6. The algorithm terminates when no more descendants are identified from the header table or if the frequencies of the newly identified nodes are less than the value of the global minimum support.

**Table 3: Action Sequence Profiles**

<table>
<thead>
<tr>
<th>Activity</th>
<th>No. of Attempts</th>
<th>No. of Profiles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video</td>
<td>89</td>
<td>10</td>
</tr>
<tr>
<td>Structured</td>
<td>286</td>
<td>11</td>
</tr>
<tr>
<td>Unstructured</td>
<td>303</td>
<td>11</td>
</tr>
<tr>
<td>MCQ</td>
<td>33</td>
<td>12</td>
</tr>
</tbody>
</table>

**5. Clustering**

Sequences of attempts for each of the four activities are clustered with the AP algorithm. The shared preferences of AP are set to the maximum of the similarity matrices. We use the R package AP for this analysis. PLWAP is then used to retrieve a descriptive summary for each cluster. We restrict the algorithm to only identify contiguous sequences.

We manually merge the clusters into ASAs based on their compositions. The compositions are determined by indicators signaling the use of certain actions between defined checkpoints, similar to the process mentioned in [10]. During the merging process for each archetype, we consider the actions spanning from the onset of a question to the first submission of the question attempt. Descriptions of the ASAs identified in the video, structured and unstructured activities are presented in Table 4, Table 5, and Table 6 respectively.

---

Figure 3: Example of a PLMS-tree for the video activity.
6. Results

6.1 Score-based Prediction of Persistence

We calculated the percentage of questions students correctly answered for the structured, unstructured, and MCQ activities. As shown in Table 7, only a student's MCQ performance is associated with persisting into Phase 2. Knowing students' performances for the structured and unstructured activities leads to a prediction accuracy level similar to that of assuming no student persists from Phase 1 to Phase 2.

As a caveat, the deterministic appearance of the association between MCQ performance and persisting to Phase 2 is misleading. The high correlation is due to the MCQ activity being a prerequisite for Phase 2. The mere presence of an MCQ submission, rather than the score itself, is predictive of persisting to Phase 2. Students who do not make an MCQ submission effectively earn a score of zero for the activity and do not have the possibility to continue to Phase 2. Additionally, all students who do persist to Phase 2 must have scored above a zero on the MCQ activity.

6.2 Sequence-based Prediction of Persistence

We converted the frequency of each ASA into a percentage of a student's total action sequences. We then used a classification and regression tree (CART) algorithm to predict which students continued on to Phase 2 based on their ASA values. The decision trees associated with progressing based on ASAs from the video, structured and unstructured activities are presented in Table 4, Table 5 and Table 6 respectively.

While persistence cannot be reliably predicted based on video ASAs, it can be accurately predicted by the structured and unstructured ASAs. The predictability of these features is
determined using a logistic regression classifier for each activity. The results are presented in Table 8.

### Table 8: Logistic regression classification for stop-out prediction.

<table>
<thead>
<tr>
<th>Variable Set</th>
<th>Variables</th>
<th>Accuracy</th>
<th>Kappa Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score-based</td>
<td>Structured</td>
<td>48.00%</td>
<td>-0.06</td>
</tr>
<tr>
<td></td>
<td>Structured + Unstructured</td>
<td>66.67%</td>
<td>0.43</td>
</tr>
<tr>
<td></td>
<td>MCQ</td>
<td>100.00%</td>
<td>1.00</td>
</tr>
<tr>
<td>Sequence-based</td>
<td>Videos</td>
<td>75.00%</td>
<td>0.48</td>
</tr>
<tr>
<td></td>
<td>Structured</td>
<td>81.48%</td>
<td>0.61</td>
</tr>
<tr>
<td></td>
<td>Unstructured</td>
<td>81.82%</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td>MCQ</td>
<td>82.35%</td>
<td>0.56</td>
</tr>
</tbody>
</table>

The stop-out prediction accuracy increases as more activity scores are included in the logistic regression models. The accuracy of these models is highly dependent on the inclusion of the MCQ activity scores. The MCQ activity is the last activity students must complete before proceeding to Phase 2.

The decision tree for the video activity, as shown in Figure 4, identify premature termination (ASA V2) as the best criterion for determining if students are likely to stop the activities. Prematurely terminating attempts at a frequency higher than 25% of the student's attempts is predictive of stopping the activity 83% of the time. In addition, a low compliance with incomplete video watching by fast-forwarding (ASA V10) and completing the video without additional actions (ASA V3) are indicative of students who stop out of the learning environment.

For the structured activity, a high compliance with the recommended process but without submitting an answer (ASA S10), answering questions with highlighting of keywords (ASA S5) and answering questions with the scripted steps, as shown in Figure 5, all indicate students who are likely to proceed to Phase 2. Students who tend not to provide an answer for these attempts are likely to not proceed to Phase 2.

While the unstructured activity gives more freedom to participants, the number of splitting criteria is minimal. Learners who do not proceed to Phase 2 are characterized by submitting more than 13% of their questions without any attempt to solve them (ASA U3). Also students who complied more than with the scripted steps more than 56% of the time also tended to stop out (ASA U12). We note that the lower compliance with the RIGHT sequence in unstructured activity in Phase 1 is associated with 86% probability of learners proceeding to Phase 2.

### 7. Conclusions

In this study, we presented a framework for converting clickstream data into action sequence archetypes. ASAs provide insight into how students approach learning activities by consolidating similar plans of action under a common label. For us, having a common label to refer to different patterns of actions facilitates discussion and interdisciplinary collaboration between the computer sciences and the learning sciences. This collaboration led us away from trying to analyze learning outcomes with click counts and time on task measures and toward ASAs. ASA frequencies identify how often a learner attempts to reach a goal via a particular method.
Looking at our decision trees, ASAs can be used to quickly identify whether a learner is using on-task or off-task behaviors. However, they also can also be used to separate different approaches to achieving the same goal.

In our case, students whose action sequences aligned more strongly to the archetype representing the RIGHT sequence presented in the online videos were less likely to persist to the second phase of activities. In one sense, it is counterintuitive to suggest that students who follow a taught script more closely would be less likely to persist in an activity. However, if script use is a way of minimizing cognitive load, novices who consistently exhibit script-like behaviors could be indicating more routinization and less assimilation of new concepts. What these students may have learned from their classroom instruction and the online material is a series of steps for completing the structured and unstructured activities and not the generalizable schema that underlies those activities.

Using the ASAs to separate script users from generalizable schema users gives us a method of predicting a student's likelihood of persisting through the first phase of activities and attempting the second phase composed of unfamiliar math models. This method of prediction identifies students who are likely to stop out before the second phase much earlier than looking at how accurately the students solve the word problems. By the end of the second activity, our model could predict with high accuracy whether a student would continue on to Phase 2. Using a more traditional method of performance assessment and analyzing accuracy levels to predict future behavior required students to complete all of Phase 1 before the model could accurately predict whether the student would persist. In short, using ASAs to analyze how students approach the activities is more diagnostic of future performance than looking at past performance measures.

Finally, it is not lost on us that we developed an algorithm that converts action sequences (scripts) into action sequence archetypes (schemas) to measure students’ use of scripts and generalizable schemas. For this project, the machine learning goals and the students learning goals happened to overlap. We plan to continue developing the parallels by integrating our ASA analysis into a student feedback engine that can shift students away from off-task behaviors and toward on-task behaviors. We also seek to lead on-task students toward more productive action sequences that foster the development of generalizable problem solving schemas rather than specific problem solving scripts.

8. ACKNOWLEDGMENTS
This project is supported by a start-up grant from the Centre for Research and Development in Learning (CRADLE@NTU).

9. REFERENCES
A Coupled User Clustering Algorithm for Web-based Learning Systems

Ke Niu∗†
kk511@bit.edu.cn
Zhendong Niu∗‡
zniu@bit.edu.cn
Can Wang‡
can.wang@csiro.au
Kai Kang∗
kangkai@bit.edu.cn
Xiangyu Zhao∗
koopr@bit.edu.cn
Can Wang‡
can.wang@csiro.au
Kai Kang∗
kangkai@bit.edu.cn
Min Ye‡
2120141074@bit.edu.cn

∗School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
†Computer School, Beijing Information Science and Technology University, Beijing, China
‡Digital Productivity, Commonwealth Scientific and Industrial Research Organisation, Sandy Bay, Australia

ABSTRACT
User clustering algorithms have been introduced to analyze users’ learning behaviors and help to provide personalized learning guides in traditional Web-based learning systems. However, the explicit and implicit coupled interactions, which means the correlations between user attributes generated from learning actions, are not considered in these algorithms. Much significant and useful information which can positively affect clustering accuracy is neglected. To solve the above issue, we proposed a coupled user clustering algorithm for Web-based learning systems. It respectively takes into account intra-coupled and inter-coupled relationships of learning data, and utilizes Taylor-like expansion to represent their integrated coupling correlations. The experiment result demonstrates the outperformance of the algorithm in terms of efficiently capturing correlations of learning data and improving clustering accuracy.

Keywords
Web-based learning, coupled interactions, user clustering, user behavior analysis

1. INTRODUCTION
Information technology and its application have brought great changes to all aspects of human, especially education area. Web-based learning is a significant and advanced way of education, meaning to utilize computer network technology, digital multimedia technology, database technology and other modern information technology to learn in digital environment. Compared with traditional learning, Web-based learning can efficiently meet learners’ needs of learning anytime and anywhere. Meanwhile, it takes advantage of various online resources and helps learners to expand their horizons and discover interests.

Recently Web-based learning systems are studied by many education institutions and researchers, and a large number of online learning communities and virtual schools arise [1]. As an emerging online learning system, MOOC (Massive Open Online Courses) was initiated by America’s top universities in 2012. It had a participation of more than 6 million of students from around 220 countries within one year [2]. Some of Web-based learning systems apply user clustering algorithms to analyze learning behaviors and provide personalized learning services. Fu and Ofoghlu put forward a new clustering algorithm; it can extract clusters which can be described by overlapping layered concept in dense space [3]. According to the feedback of basic clustering method, Montazer et al. proposed a hybrid clustering algorithm, which considered clustering issues from different perspectives, and kept the simplicity of basic clustering algorithm [4]. Another matrix-based improved clustering algorithm was put forward by Zhang et al., and it is much more efficient when comparing with K-means [5]. Lin et al. came up with a kind of intuitionistic fuzzy kernel clustering algorithm (KIFCM), combining intuitionistic fuzzy sets and fuzzy kernel clustering algorithm, and applied it in learner behavior analysis [6].

With the above algorithms utilized in Web-based learning systems, learners’ attribute information is extracted by analyzing their behaviors, and finally used for user clustering. However, these algorithms generally neglect the explicit and implicit coupling relationships of user attributes and this may lead to massive significant information loss. For example, table 1 presents an evaluation index system based on information provided by a specified Web-based learning system. With common sense, we think that user attribute of “Average correct rate of homework” has a positive impact on “Comprehensive test result”. Generally, if the “Average correct rate of homework” is better, the “Comprehensive test result” is better. Students who behave this way are categorized in “normal” group. However, there are also students who can either get a better “Average correct rate of homework” with a worse “Comprehensive test result”, or a better “Comprehensive test result” with a worse “Average correct rate of homework”; they are categorized in “unnormal” group. These unnormal situations are caused by irregular correlations of user attributes, but they are often ignored. This will certainly have negative effect on user clustering.
Table 1: Comprehensive evaluation index system

<table>
<thead>
<tr>
<th>First-level index</th>
<th>Second-level index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autonomic learning</td>
<td>Times of doing homework</td>
</tr>
<tr>
<td></td>
<td>Average correct rate of homework</td>
</tr>
<tr>
<td></td>
<td>Number of learning resources</td>
</tr>
<tr>
<td></td>
<td>Total time length of learning resources</td>
</tr>
<tr>
<td></td>
<td>Times of daily quiz</td>
</tr>
<tr>
<td></td>
<td>Daily average quiz result</td>
</tr>
<tr>
<td></td>
<td>Comprehensive test result</td>
</tr>
<tr>
<td></td>
<td>Number of collected resources</td>
</tr>
<tr>
<td></td>
<td>Times of downloaded resources</td>
</tr>
<tr>
<td></td>
<td>Times of making notes</td>
</tr>
<tr>
<td></td>
<td>Times of asking questions</td>
</tr>
<tr>
<td></td>
<td>Times of answering classmates’ questions</td>
</tr>
<tr>
<td></td>
<td>Times of posting comments on the BBS</td>
</tr>
<tr>
<td></td>
<td>Times of interaction by BBS message</td>
</tr>
<tr>
<td>Interactive learning</td>
<td>Average marks made by the teacher</td>
</tr>
<tr>
<td></td>
<td>Average marks made by other students</td>
</tr>
<tr>
<td></td>
<td>Times of marking and remarking made by the student for the teacher</td>
</tr>
<tr>
<td></td>
<td>Times of marking and remarking made by the student for other students</td>
</tr>
</tbody>
</table>

Nowadays an increasing number of researchers are studying the interactions between object attributes with special attention and have been aware that the independence assumption on attributes often leads to a mass of information loss. In addition to the basic Pearson’s correlation [7], Wang et al. put forward the intra-coupled and inter-coupled interactions of continuous attributes [8]. An innovative coupled group-based matrix factorization model for discrete attributes of recommender system was addressed by Li et al. [9]. Jakulin and Bratko proposed an algorithm to detect interactions between attributes, but it is only applicable in supervised learning with the experimental results [10]. For unsupervised learning, the coupled nominal similarity to extract new relationships between entities was addressed by Wang et al., but it is only for categorical data [11]. We rarely find any methods applied in Web-based learning systems, that consider coupling relationships of user attributes in user clustering.

This paper proposed a coupled user clustering algorithm for Web-based learning systems, namely CUCA. It studies the coupling relationships of user attributes. With the help of Taylor-like expansion, we use a spectral clustering algorithm to cluster users. When it is applied in Web-based learning systems, it can efficiently capture learners’ behavior features and analyze the information behind them, especially that of “unnormal” group of learners, and finally use them to provide personalized learning services. To verify the outperformance of CUCA, we compare its clustering result with that of 3 other algorithms, respectively from 3 dimensions of learning attitude, learning effect and the integrated dimension.

The rest of the paper is organized as following. The clustering algorithm model is proposed in section 2. Section 3 introduces the formalization and exemplification of the clustering algorithm. In section 4, experiments and results analysis are demonstrated. Section 5 concludes the paper and discusses some potential applications of the proposed algorithm in the future.

2. CLUSTERING MODEL

Evaluation model usually plays the core role in user evaluation framework [12]. In this section, the coupled user clustering model is illustrated. This model captures coupling relationships of user attributes through online behavior analysis, and uses spectral clustering algorithm to improve clustering accuracy.

![Figure 1: The coupled user clustering model](image)

The model is composed of user learning behavior analysis, coupled interactions computation of user attributes, integrated coupling representation and spectral clustering algorithm, illustrated in figure 1. As the basis, data for user learning behavior analysis needs to be collected, consolidated and normalized. From the data, user attributes information is extracted. With the extracted user attributes, the intra-coupled interaction within an attribute and inter-coupled interaction among different attributes are respectively captured. Then all the interactions are integrated and represented using Taylor-like expansion. Finally we use a spectral clustering algorithm - NJW to cluster users. This model is consequently applied in various Web-based personalized services, like Learning guide customization, tutoring and learning resources recommendation.

3. CLUSTERING ALGORITHM

Based on the model illustrated in section 2, this paper proposed an online coupling user clustering algorithm. It is
Table 2: A fragment example of user attributes

<table>
<thead>
<tr>
<th>U</th>
<th>A</th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
</tr>
</thead>
<tbody>
<tr>
<td>u1</td>
<td>0.61</td>
<td>0.55</td>
<td>0.47</td>
<td>0.72</td>
<td>0.63</td>
<td>0.62</td>
<td></td>
</tr>
<tr>
<td>u2</td>
<td>0.75</td>
<td>0.92</td>
<td>0.62</td>
<td>0.63</td>
<td>0.74</td>
<td>0.74</td>
<td></td>
</tr>
<tr>
<td>u3</td>
<td>0.88</td>
<td>0.66</td>
<td>0.71</td>
<td>0.74</td>
<td>0.85</td>
<td>0.87</td>
<td></td>
</tr>
<tr>
<td>u4</td>
<td>0.24</td>
<td>0.83</td>
<td>0.44</td>
<td>0.29</td>
<td>0.21</td>
<td>0.22</td>
<td></td>
</tr>
<tr>
<td>u5</td>
<td>0.93</td>
<td>0.70</td>
<td>0.66</td>
<td>0.81</td>
<td>0.95</td>
<td>0.93</td>
<td></td>
</tr>
</tbody>
</table>

suitable for network education, not only applicable to user clustering analysis in Web-based learning systems, but also to enterprise training, performance review and others with users participation and behaviors recording. This section describes the details of the proposed coupled user clustering algorithm. Firstly, it collects user learning behavior information and extracts user attributes from them. Secondly, it calculates and represents users’ intra-coupled and inter-coupled relationship. Thirdly, the intra-coupled and inter-coupled interactions are integrated to be a coupled representation. Finally, it clusters users based on the processed attributes, using NJW algorithm.

3.1 User learning behavior analysis

When students login a Web-based learning system, the system will record their activity information, such as number of learning resources, total time length of learning resources and average correct rate of homework, which can be used to build an evaluation index system. We refer to a Web-based personalized user evaluation model [13] and utilizes its evaluation index system to extract students’ attributes information. This index system is with evaluation standards of America K-12 (kindergarten through twelfth grade) [14] and Delphi method [15], which is a hierarchical structure built according to mass of information and data generated during general e-learning activities. It defines 20 indicators and can comprehensively represent the students’ attributes, as shown in table 1.

Generally attributes are with various data types and units, we formalize them by creating the table 2.

3.2 Intra-coupled and inter-coupled representation

In this section, we represent intra-coupled and inter-coupled interactions of user attributes. And with a few examples, the application of CUCA is demonstrated. We choose 5 students and 6 of the 20 attributes in table 1, which are “Average correct rate of homework”, “Times of doing homework”, “Number of learning resources”, “Total time length of learning resources”, “Daily average quiz result” and “Comprehensive test result”. The 6 attributes are respectively signified by a1, a2, a3, a4, a5 and a6 in table 2.

Here we use a tetrad $S = \langle U, A, V, f \rangle$ to represent user attributes information. $U = \{u_1, u_2, \ldots, u_m\}$ means a finite set of users; $A = \{a_1, a_2, \ldots, a_n\}$ refers to a finite set of attributes; $V = \bigcup_{j=1}^{n} V_j$ represents all attributes value sets; $V_j = \{a_{j_1}, a_{j_2}, \ldots, a_{j_l}, v_{j_k}\}$ is the value set of the $j$-th attribute; $f = \bigcup_{j=1}^{n} f_j, f_j : U \rightarrow V_j$ is the function for calculating a certain attribute value. For example, the information table 2 above contains 5 users $\{u_1, u_2, u_3, u_4, u_5\}$ and 6 attributes $\{a_1, a_2, a_3, a_4, a_5, a_6\}$; the first attribute value of $u_1$ is $f_1(u_1) = 0.61$.

The common way to calculate the interactions between 2 attributes is Pearson’s correlation coefficient [7]. The user attributes from the Table 1 are continuous variables and approximate to Normal distribution, meeting the constraint condition of the Pearson’s correlation coefficient. Thus we use it to help to calculate attributes interactions in this paper. For instance, the Pearson’s correlation coefficient between $a_k$ and $a_j$ is formalized as:

$$
Cor(a_j, a_k) = \frac{\sum_{u \in U} (f_j(u) - \mu_j)(f_k(u) - \mu_k)}{\sqrt{\sum_{u \in U} (f_j(u) - \mu_j)^2 \sum_{u \in U} (f_k(u) - \mu_k)^2}}
$$

(1)

Where $\mu_j, \mu_k$ are respectively means of $a_j, a_k$.

The Pearson’s correlation coefficient helps to calculate the attributes interactions, but it fits for linear relationship only, which is not sufficient to fully capture pairwise attributes interactions. Therefore we converts the original data attributes into a higher dimensional feature space to extract more attribute information [16].

Firstly, we use a few additional attributes to expand interaction space. Then there are $L$ attributes for each original attribute $a_i$, including itself, namely $(a_i)^1, (a_i)^2, \ldots, (a_i)^L$. Each attribute value is the power of the attribute, for instance, $(a_i)^3$ is the third power of attribute $a_i$, $(a_i)^p (1 \leq p \leq L)$ is the $p$-th power of $a_i$. In table 3, the denotation $a_j$ and $(a_j)^p$ are equivalent; the value of $(a_j)^2$ is the square of that of $a_j$. For simplicity, we set $L=2$.

Secondly, the correlation between pairwise attributes is calculated. It captures both local and global coupling relations. We take the $p$-values for testing the hypotheses of no correlation between attributes into account. $p$-value here means the probability of getting the maximum correlation observed by random chance, while the true correlation is zero. If $p$-value is smaller than 0.05, the correlation $Cor(a_j, a_k)$ is significant. The updated correlation coefficient is as:

$$
R_Cor(a_j, a_k) = \begin{cases} 
Cor(a_j, a_k) & \text{if } p\text{-value} < 0.05, \\
0 & \text{otherwise}.
\end{cases}
$$

(2)

Here we do not consider all relationships, but only takes the significant coupling relationships into account, because all relationships involvement may cause the over-fitting issue on modeling coupling relationship. This issue will go against the attribute inherent interaction mechanism. So based on the updated correlation, the intra-coupled and inter-coupled interaction of attributes is proposed. Intra-coupled interaction is the relationship between $a_j$ and all its powers; inter-coupled interaction is the relationship between $a_j$ and powers of the rest attributes $a_k (k \neq j)$.
Table 3: Extended user attributes

<table>
<thead>
<tr>
<th></th>
<th>(a_1)</th>
<th>(a_2)</th>
<th>(a_3)</th>
<th>(a_4)</th>
<th>(a_5)</th>
<th>(a_6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(u_1)</td>
<td>0.61</td>
<td>0.37</td>
<td>0.55</td>
<td>0.30</td>
<td>0.47</td>
<td>0.22</td>
</tr>
<tr>
<td>(u_2)</td>
<td>0.75</td>
<td>0.56</td>
<td>0.92</td>
<td>0.85</td>
<td>0.62</td>
<td>0.38</td>
</tr>
<tr>
<td>(u_3)</td>
<td>0.88</td>
<td>0.77</td>
<td>0.66</td>
<td>0.44</td>
<td>0.71</td>
<td>0.50</td>
</tr>
<tr>
<td>(u_4)</td>
<td>0.24</td>
<td>0.06</td>
<td>0.83</td>
<td>0.69</td>
<td>0.44</td>
<td>0.19</td>
</tr>
<tr>
<td>(u_5)</td>
<td>0.93</td>
<td>0.86</td>
<td>0.70</td>
<td>0.49</td>
<td>0.66</td>
<td>0.44</td>
</tr>
</tbody>
</table>

Definition 1 Intra-coupled interaction. The intra-coupled interaction within an attribute is represented as a matrix. For attribute \(a_j\), it is an \(L \times L\) matrix \(R^{a_j}\). In the matrix, \((p,q)\) is the correlation between \(a_j^p\) and \(a_j^q\) \((1 \leq p,q \leq L)\).

\[
R^{a_j} = \begin{pmatrix}
\alpha_{11}(j) & \alpha_{12}(j) & \cdots & \alpha_{1L}(j) \\
\alpha_{21}(j) & \alpha_{22}(j) & \cdots & \alpha_{2L}(j) \\
\cdots & \cdots & \cdots & \cdots \\
\alpha_{L1}(j) & \alpha_{L2}(j) & \cdots & \alpha_{LL}(j)
\end{pmatrix}
\]

(3)

Where \(\alpha_{pq}(j) = R^{a_j}\text{Cor}(a_j^p, a_j^q)\) is the Pearson’s correlation coefficient between \(a_j^p\) and \(a_j^q\).

For attribute \(a_1\) in table 3 above, we can get the intra-coupled interaction of it as \(R^{a_1}(a_1) = \begin{pmatrix} 1 & 0.986 \\ 0.986 & 1 \end{pmatrix}\), which means that the correlation coefficient between attribute “Average correct rate of homework” and its second power is as high as 0.986. There is close relationship between them.

Definition 2 Inter-coupled interaction. The inter-coupled interaction between attribute \(a_j\) and other attributes \(a_k\) \((k \neq j)\) is quantified as an \(L \times L \times (n-1)\) matrix as:

\[
R^{a_j|\{a_k\}_{k \neq j}} = \left( R^{a_j|a_{k_1}} \cdots R^{a_j|a_{n-1}} \right)
\]

(4)

\[
R^{a_j|a_k} = \begin{pmatrix}
\beta_{11}(j|k) & \beta_{12}(j|k) & \cdots & \beta_{1L}(j|k) \\
\beta_{21}(j|k) & \beta_{22}(j|k) & \cdots & \beta_{2L}(j|k) \\
\cdots & \cdots & \cdots & \cdots \\
\beta_{L1}(j|k) & \beta_{L2}(j|k) & \cdots & \beta_{LL}(j|k)
\end{pmatrix}
\]

(5)

Here \(\{a_k\}_{k \neq j}\) refers to all the attributes except for \(a_j\), and \(\beta_{pq}(j|k) = R^{a_j}\text{Cor}(a_j^p, a_k^q)\) is the correlation coefficient between \(a_j^p\) and \(a_k^q\) \((1 \leq p,q \leq L)\).

For attribute \(a_1\) in the table 3 above, the inter-coupled interaction between \(a_1\) and others \((a_2, a_3, a_4, a_5, a_6)\) is calculated as:

\[
R^{a_1|\{a_2, a_3, a_4, a_5, a_6\}} = \begin{pmatrix} 0 & 0 & 0.898 & 0.885 & 0.928 & 0.921 \\
0 & 0 & 0.929 & 0.920 & 0.879 & 0.888 \\
0 & 0 & 0.997 & 0.982 & 0.999 & 0.988 \\
0 & 0 & 0.978 & 0.994 & 0.982 & 0.999 \\
0 & 0 & 0.003 & 0 & 0.002 & 0
\end{pmatrix}
\]

The p-values between \(a_1\) and others \((a_2, a_3, a_4, a_5, a_6)\) is calculated as:

\[
p^{a_1|\{a_2, a_3, a_4, a_5, a_6\}} = \begin{pmatrix} 0.689 & 0.677 & 0.039 & 0.046 & 0.023 & 0.027 \\
0.733 & 0.707 & 0.023 & 0.027 & 0.050 & 0.044 \\
0.004 & 0.001 & 0.003 & 0 & 0 & 0
\end{pmatrix}
\]

Based on the result, we can find that there is hidden correlation between user attributes. For instance, all the p-values between attribute \(a_1\) and \(a_2\) are larger than 0.05, so the correlation coefficient is 0 based on Equation (2), indicating there is no significant correlation between “Average correct rate of homework” and “Times of doing homework”. Meanwhile, the correlation coefficient between \(a_1\) and \(a_5\) and \(a_6\) is quite close to 1; it indicates “Daily average quiz result” and “Comprehensive test result” respectively have close relationship with “Average correct rate of homework”, which is consistent with our practical experiences. In conclusion, comprehensively taking into account intra-coupled and inter-coupled correlation of attributes can efficiently help capturing coupling relationships between user attributes.

3.3 Integrated coupling representation

Intra- and inter-coupled interactions are integrated in this section as a coupled representation scheme.

In table 3 above, each user is signified by \(L \times n\) updated variables \(\tilde{A} = \{a_1\}^L, \ldots, \{a_{n}\}^L\). With the updated function \(T^u_j(u)\), the corresponding value of attribute \(a_1^p\) is assigned to user \(u\). Attribute \(a_j\) and all its powers are signified as \(\tilde{u}(a_j) = T^u_1(u), \ldots, T^u_j(u)\), while the rest attributes and all powers are presented in another vector \(\tilde{u}(\{a_k\}_{k \neq j}) = \{\tilde{T}_1(u), \ldots, \tilde{T}_{j-1}(u), \tilde{T}_{j+1}(u), \ldots, \tilde{T}_{n-1}(u)\}\). For instance, in table 3, \(\tilde{u}(a_1) = [0.61, 0.37, 0.55, 0.30, 0.47, 0.22, 0.72, 0.52, 0.63, 0.40, 0.02, 0.35]\).

Definition 3 Coupled representation. Attribute \(a_j\)’s coupled representation is formalized as a \(1 \times L\) vector \(u^c(a_j|\tilde{A}, L)\),
where (1, p) component corresponds to the updated attribute \((a_j)^p\).

\[
\alpha^p(j,\tilde{A},L) = \frac{\beta_l(j|k_{ij})}{\sum_{l=1}^{L} \beta_l(j|k_{ij})} k_{ij}(u) + \frac{\beta_l(j|k_{ij})}{2!} \tilde{k}_{ij}(u) + \frac{\beta_l(j|k_{ij})}{3!} \frac{\tilde{k}_{ij}(u)}{2} + \ldots
\]

Finally we obtained the global coupled representation of all the \(n\) original attributes as a concatenated vector:

\[
u^c(\tilde{A},L) = [u^c(a_1,\tilde{A},L), u^c(a_2,\tilde{A},L), \ldots, u^c(a_n,\tilde{A},L)]^T
\]

With the couplings of attributes, each user is represented as a \(1 \times L \times n\) vector. When all the users follow the steps above, we then obtain an \(m \times L \times n\) coupled information table. For example, based on Table 2, the coupled information table shown in Table 4, is the new representation.

### 3.4 User clustering

We obtained the global coupled representation in Table 4. Compared with the original representation, this one reflects coupling interactions of attributes, and contains far more coupling relationships. With these data, we can do user clustering using NJW [18], which is a kind of spectral clustering algorithm. Detailed clustering results are demonstrated in experiment later.

### 4. EXPERIMENTS AND EVALUATION

In this section, we conduct experiments to verify the validity and accuracy of the proposed algorithm. The data for the experiments are collected from a Web-based learning system of China Educational Television (CETV), named "New Media Learning Resource Platform for National Education". As a basic platform for national lifelong education, which started the earliest in China, and had the largest group of users and provided most extensive learning resources, it met the needs of personalization and user diversity through integrating a variety of multi-network, terminals and resources. So far, the number of registered users has reached more than two million. The experiment is composed of 3 parts: user study, user clustering and result analysis.

### 4.1 User study

In the experiment, we ask 220 users (signified by \(s_1, s_2, \ldots, s_{220}\)) to learn \(C\) programming language online. The whole learning process, including recording and analyzing learning activities information, is accomplished in CETV.

The public data sets regarding learners’ learning behaviors in online learning systems are insufficient, and most of them don’t contain labeled user clustering information. Meanwhile, because learners always behave with certain subjectivity in online learning systems, to label learners with different classifiers based on their learning behaviors only, but without the information behind, is not accurate. Therefore, we adopt a few user study methods, including self-assessment, peer-assessment and teacher-assessment [19], to label online learners with classifiers. It is the basis for verifying the accuracy of clustering.

Analyzing the 20 attributes extracted from Table 1 using user evaluation index system proposed in this paper, we can easily find that they can be mainly divided into 2 categories. Some attributes belong to the category of “learning attitude”, which refers to students’ learning initiatives, like “Times of doing homework”, “Number of learning resources” and “Total time length of learning resources”. While the rest belong to the category of “learning effect”, which refers to how well students receive knowledge, like “Average correct rate of homework”, “Daily average quiz result” and “Comprehensive test result”. Accordingly, we can label learners with these attributes from both categories. Each of the attributes has 3 grades - high, medium and low. Consequently every learner has 2 labels and each label has one grade of high, medium and low. In total, there will be 9 different combinations - high & high, high & medium, high & low, medium & high, medium & medium, medium & low, low & high, low & medium and low & low.

After the students had finished a learning phase, we asked the 220 users to do a self-assessment using centesimal grade, respectively from perspectives of learning attitude and learning effect. Then we requested teacher assessments in the same way, meaning the teacher of the subject to review the students’ performance. Finally, the students were asked to do peer-assessments, which means students do an assessment for each other. Each student will get the assessment scores from the rest 219 students. We calculate the aver-

---

1http://www.guoshi.com/
Table 4: Integrated coupling representation of user attributes

<table>
<thead>
<tr>
<th>U</th>
<th>\langle a_1 \rangle^1 \langle a_1 \rangle^2</th>
<th>\langle a_2 \rangle^1 \langle a_2 \rangle^2</th>
<th>\langle a_3 \rangle^1 \langle a_3 \rangle^2</th>
<th>\langle a_4 \rangle^1 \langle a_4 \rangle^2</th>
<th>\langle a_5 \rangle^1 \langle a_5 \rangle^2</th>
<th>\langle a_6 \rangle^1 \langle a_6 \rangle^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>u_1</td>
<td>3.85 3.80 0.70 0.70 2.20 1.46 3.24 3.23 3.35 3.70 3.76 3.81</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>u_2</td>
<td>4.54 4.50 1.34 1.34 2.89 1.98 3.66 3.65 3.82 4.31 4.37 4.51</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>u_3</td>
<td>5.51 5.46 0.88 0.88 3.54 2.44 4.46 4.45 4.66 5.22 5.28 5.47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>u_4</td>
<td>1.53 1.52 1.17 1.17 1.01 0.80 1.03 1.02 1.06 1.42 1.44 1.52</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>u_5</td>
<td>5.94 5.89 0.94 0.94 3.73 2.49 4.95 4.94 5.17 5.68 5.75 5.90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Transformation rule between score and grade

<table>
<thead>
<tr>
<th>Score range</th>
<th>Grade</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>80 ≤ X ≤ 100</td>
<td>high</td>
<td>95</td>
</tr>
<tr>
<td>50 ≤ X &lt; 80</td>
<td>medium</td>
<td>75</td>
</tr>
<tr>
<td>0 ≤ X &lt; 50</td>
<td>low</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 6: The evaluation results of \( s_1 \)

<table>
<thead>
<tr>
<th>learning attitude</th>
<th>learning effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-assessment (40%)</td>
<td>80.0 75.0</td>
</tr>
<tr>
<td>Teacher-assessment (35%)</td>
<td>85.0 80.0</td>
</tr>
<tr>
<td>Peer-assessment (25%)</td>
<td>82.7 79.2</td>
</tr>
<tr>
<td>Comprehensive evaluation results</td>
<td>82.4 77.8</td>
</tr>
<tr>
<td>grade</td>
<td>high medium</td>
</tr>
</tbody>
</table>

The transformation rule between score and grade is shown in Table 5.

Take student \( s_1 \) as an example, his 3 assessment scores and transformed grades are shown in Table 6.

4.2 User clustering

In Equation (9), the proposed coupled representation is strongly dependent on how large \( L \) can be. Thus we conduct a few experiments to study how the performance of \( L \) influences the clustering accuracy of CUCA. The range of \( L \) value is from \( L = 1 \) to \( L = 10 \). With the growth of \( L \) value, \( L \) value grows. When \( L = 10 \), it is large enough to capture most of the information in Equation (9). The experiments show that with the growth of \( L \), the clustering accuracy will be gradually improved. When \( L = 3 \), the accuracy change reaches a comparatively stable status; when \( L > 3 \), the accuracy change is extremely small. That means the accuracy of when \( L = 3 \) and when \( L = 10 \) is quite similar. To guarantee the accuracy of experimental results and reduce the complexity of the algorithm, we take \( L = 3 \) in the following comparative experiments.

In the experiments, we utilize the attributes data generated from the 220 students’ learning process, as the basis for clustering. Then we persistently collect data from the process which reaches 30 hours by average. Respectively with the help of K-means algorithm, Fuzzy C-means algorithm (FCM), NJW algorithm and CUCA algorithm, we do user clustering, getting 2 labels in terms of learning attitude and learning effect for each student. In section 4.1, we classified each student with 2 labels based on user study result. Then we compare the labels got from user study and user clustering result. If only one label from each side is the same, the clustering accuracy rate is 50%; if both the labels are the same, the accuracy rate reaches 100%. For instance, student \( s_1 \) is labeled with “high & medium” in user study, if he is classified to “medium & medium” cluster, the clustering accuracy rate is 50%; if he is classified to “high & medium” cluster, the accuracy rate reaches 100%.

4.3 Result analysis

We do comparison analysis on the clustering result respectively from the 3 dimensions of learning attitude, learning effect and the integrated dimension. The analysis result is shown in Figure 2. We can see the clustering accuracy of utilizing CUCA is 89.4% for learning attitude, 87.3% for learning effect and 74.6% for integrated dimension, each of which is higher than that with the other 3 algorithms. Especially, CUCA obviously outperforms the rest on clustering accuracy of integrated dimensions. Compared with K-means which performs the worst, CUCA improves almost 30% on
the clustering accuracy. The reason is CUCA fully takes into account coupling relationships of users. In Web-based learning systems, if the user attributes are more complicated, there will be more clustering dimensions and the clustering accuracy will be improved more.

Besides, we can verify clustering accuracy through analyzing user clustering results. The best performance of a clustering algorithm is keeping the distance within clusters as small as possible and the distance between clusters as large as possible. We use the evaluation criteria of Relative Distance (the ratio of average inter-cluster distance upon average intra-cluster distance) and Sum Distance (the sum of object distances within all the clusters) to present the distance. The larger Relative Distance is and the smaller Sum Distance is, the better clustering results are. From figure 4, we can see that the Relative Distance for CUCA is larger than that of the 3 other algorithms, while the Sum Distance for CUCA is smaller. It indicates that CUCA outperforms the rest in terms of clustering structure.

Figure 3: Clustering result of different time phases

If we divide the process of extracting user attributes to 6 phases, namely 5h, 10h, 15h, 20h, 25h, 30h based on average learning length, we can get the correlation between average learning length and clustering accuracy, as shown in figure 3. From the figure, we can see that while the learning length grows, the clustering accuracy of the 4 algorithms keeps improving, specifically for CUCA. With CUCA, the clustering accuracy on integrated dimensions distinctly outperforms that of the 3 other algorithms. It indicates that with the increasing learning behavior data volume, CUCA can find the hidden coupling relationships of user attributes more easily, and the clustering accuracy is much better.

Figure 4: Clustering structure analysis (30h)

5. CONCLUSION

A coupled user clustering algorithm (CUCA) for Web-based learning systems is proposed in this paper to capture coupling relationships of user attributes. The algorithm respectively takes intra-coupled and inter-coupled correlation into account in the application process, and utilizes Taylor-like expansion to represent the coupling relationship. Finally, with the usage of spectral clustering algorithm, CUCA is applied to do user clustering. In the experiments, user study, user clustering and result analysis are adopted to verify that CUCA outperforms traditional algorithm for user clustering.

In this paper, the user attributes extracted from user learning behavior data are all numerical data, most of which are continuous data. In reality, there are also categorical data, which will be a significant study topic in the future.
6. ACKNOWLEDGMENTS
This work is supported by the National Natural Science Foundation of China (Project No. 61370137), the National “973” Project of China (No. 2012CB720702) and Major Science and Technology Project of Press and Publication (No: GAPP_ZDKJ_BQ/01).

7. REFERENCES
Execution Traces as a Powerful Data Representation for Intelligent Tutoring Systems for Programming

Benjamin Paaßen
CITEC Center of excellence
Inspiration 1
33619 Bielefeld, Germany
bpaassen@techfak.uni-bielefeld.de

Joris Jensen
CITEC Center of excellence
Inspiration 1
33619 Bielefeld, Germany
jjensen@techfak.uni-bielefeld.de

Barbara Hammer
CITEC Center of excellence
Inspiration 1
33619 Bielefeld, Germany
bhammer@techfak.uni-bielefeld.de

ABSTRACT
The first intelligent tutoring systems for computer programming have been proposed more than 30 years ago, mostly focusing on well-defined programming tasks e.g. in the context of logic programming. Recent systems also teach complex programs, where explicit modelling of every possible program and mistake is no longer possible. Such systems are based on data-driven approaches, which focus on the syntax of a program or consider the output for example cases. However, the system’s understanding of student programs could be enriched by a deeper focus on the actual execution of a program. This requires a suitable data representation which encodes information of programming style as well as its functionality in a suitable way, thus offering entry points for automated feedback generation.

In this contribution we propose a representation of computer programs via execution traces for example input and demonstrate the power of this representation in three key challenges for intelligent tutoring systems: identifying the underlying solution strategy, identifying erroneous solutions and locating the errors in erroneous programs for feedback display.

Keywords
execution traces, data-driven tutoring systems, computer science teaching, sequence alignment, sorting programs

1. INTRODUCTION
Teaching computer programming has been a long-standing goal of intelligent tutoring systems research. The earliest example, the LISP tutor, has been released in 1985 [1] and since then many different approaches have evolved, such as learning by examining and manipulating examples, by simulation and debugging, by dialogue with the system, by collaboration with peers or by feedback [7]. Most of these approaches rely on extensive domain knowledge about program structure, typical mistakes (so-called buggy rules) and syntactic concepts, which is expensive to obtain and difficult to encode [5, 10]. In particular, such approaches get infeasible if the space of possible programs (and mistakes) gets too large, and if the goal of the computer program is ill-defined [8]. To push the boundaries of intelligent tutoring systems towards such scenarios, data-driven approaches have been developed which provide feedback to students based on example programs handed in by other students, e.g. by highlighting the difference of the student solution and a similar, correct program [2, 16]. However, such approaches focus strongly on the syntax of programs, which is problematic because the relation between a program’s functionality and its syntax is highly non-linear.

As an example, consider the Java code shown in Figure 1. The programs on the left and on the middle are both (correct) sorting programs, which have a very similar syntactic structure. Both sort the array via two nested loops, compare the current element to its successor and swap them if the order is incorrect. However, the programs implement different algorithms, namely BubbleSort (left) and InsertionSort (middle). Thus, minor syntactic changes correspond to major changes in terms of function [14]. If an intelligent tutoring system provides feedback based on a functionally dissimilar example (e.g. a different underlying algorithm) the system might recommend changes to the student’s program which lead the learner away from her intended strategy. Such feedback might be detrimental to the student’s learning success.

This poses a challenge to educational datamining research. How do we estimate the similarity between programs on a functional level, without exceeding effort in knowledge engineering? We propose to represent computer programs by their execution traces, to compare such traces using sequence alignment and to define the similarity between programs based on the alignment distance. An execution trace is a sequence of variable states for each step of the program’s execution for some input. They are a usual representation of computer programs for debugging purposes and can provide insight into the dynamic behaviour of programs [6]. In particular, traces and alignments of traces have been successfully applied in educational programming environments to offer students an alternative view on their own program for self-reflection [17, 18]. We build upon this research by utilizing the trace representation for educational datamining,
public static int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    if (A[i] > A[i - 1]) {
      final int tmp = A[i - 1];
      A[i - 1] = A[i];
      A[i] = tmp;
    }
    insertionSort(A, l, r - 1);
  }
  return A;
}

private static void insertionSort(int[] A, int l, int r) {
  if (A[r - 1] > A[r]) {
    final int tmp = A[r - 1];
    A[r - 1] = A[r];
    A[r] = tmp;
  } }

private static void insert(int[] A, int l, int r) {
  if (A[r - 1] > A[r]) {
    final int tmp = A[r - 1];
    A[r - 1] = A[r];
    A[r] = tmp;
  } 
}

Figure 1: Three correct sorting programs in Java code. Important syntactic constructs and variable initializations are highlighted. The corresponding code parts between all three programs are visualized via background highlighting. Left: An iterative BubbleSort implementation. Middle: An iterative InsertionSort implementation. Right: A recursive InsertionSort implementation.

<table>
<thead>
<tr>
<th>Bubble</th>
<th>Insertion</th>
<th>recursive</th>
</tr>
</thead>
<tbody>
<tr>
<td>4, 7, 2, 1</td>
<td>4, 7, 2, 1</td>
<td>4, 7, 2, 1</td>
</tr>
<tr>
<td>4, 2, 7, 1</td>
<td>4, 2, 7, 1</td>
<td>4, 2, 7, 1</td>
</tr>
<tr>
<td>4, 2, 1, 7</td>
<td>2, 4, 7, 1</td>
<td>2, 4, 7, 1</td>
</tr>
<tr>
<td>2, 4, 1, 7</td>
<td>2, 4, 1, 7</td>
<td>2, 4, 1, 7</td>
</tr>
<tr>
<td>2, 1, 4, 7</td>
<td>2, 1, 4, 7</td>
<td>2, 1, 4, 7</td>
</tr>
<tr>
<td>1, 2, 4, 7</td>
<td>1, 2, 4, 7</td>
<td>1, 2, 4, 7</td>
</tr>
</tbody>
</table>

Table 1: The execution traces for the three programs from Figure 1 for the input array A = [4, 7, 2, 1]. Only the values for the variable A are shown and intermediate steps that do not manipulate A have been omitted.

that is, for automated classification and analysis of student’s computer programs in order to provide helpful, automated feedback.

As an example, consider the programs from Figure 1 again. Their execution traces for the input array A = [4, 7, 2, 1] are shown in Table 1. Despite the apparent syntactic similarity, the implementations of BubbleSort and InsertionSort do indeed map to different traces, while the iterative and recursive implementation of InsertionSort map to the same trace. This indicates that traces have a more direct relationship to the semantics of the underlying program, making them a promising representation for intelligent tutoring systems.

The main contributions of our work are as follows: First, we introduce execution traces with the purpose to capture syntactic as well as semantic aspects of the underlying program (Section 3). Second, we provide an efficient methodology for automatically comparing such traces via edit distances and inferring a measure of similarity for further datamining applications (Section 4). Finally, we evaluate our approach in comparison with the state of the art in syntactic representation in three key challenges for educational data mining: 1.) identifying the student’s underlying algorithmic approach (Section 5.2), 2.) identifying erroneous implementations (Section 5.3), and 3.) detecting the location of errors for feedback (Section 5.4). To our knowledge, no data-driven approach exists to date which tackles all three challenges. Syntax-based representations have been successful in identifying the programming strategy [11, 13] but fail in identifying erroneous solutions as well as error locations (as we will show later). On the other hand, test case-based evaluations are very successful in identifying erroneous solutions but treat programs as a black box and thus can make no claims regarding the implemented strategy or the location of the error [17].

2. BACKGROUND AND RELATED WORK

2.1 Tutoring Systems for Computer Programming

In a review of AI-supported tutoring approaches for computer programming, Le and colleagues found six categories of approaches, namely: 1.) displaying examples of programs in order to learn to construct programs of a similar type or modify examples; 2.) simulating the execution of a program in a micro-world and visualizing it to the user; 3.) providing a dialogue environment in order to complete a programming task in an interactive dialogue with the system; 4.) presenting buggy example code in order to learn via program analysis and debugging; 5.) providing feedback to students during development of their program in order to guide them towards a correct solution and detect errors; and finally 6.) providing a collaborative work environment in which students can help each other in developing a program, guided by the system’s group model [7]. We note that Le and colleagues do not yet consider recent data-driven approaches, which are mostly feedback-based systems, such as the FIT Java Tutor [2], BOTS [4] and ITAP [16]. Our own approach is targeted mainly at such feedback-based systems working on examples. We analyze the execution trace of a student’s program in order to find similar programs for feedback purposes and we intend to locate errors in the student’s program to help her correct them. However, our approach also bears similarity to simulation-based approaches as we consider the execution of the program’s statements as the main characteristic of a program.
2.2 Representations of Computer Programs for Data-Driven Systems

Most existing data-driven systems for computer programming represent programs as abstract syntax trees, which are subjected to some form of canonization in order to abstract from mere stylistic differences [15]. Recently, Piech and colleagues have criticized this approach and judged syntax trees not sufficiently discriminative to capture the strong functional consequences of small syntactic changes [14]. Instead, they propose a neural network-based approach to infer a vectorial representation of programs, such that standard machine learning methods can be applied in the resulting Euclidean space. Similar to our approach, Piech and colleagues intend to represent a programs function (or semantics) in opposition to its syntax. However, they focus on a direct mapping between input and output of program segments, while the trace representation provides more procedural (or dynamic) insight into the programs function.

2.3 Edit Distances on Computer Programs

Computing similarities and dissimilarities between computer programs is a crucial step towards data-driven intelligent tutoring system [9]. Edit distances have been particularly prominent in this regard. For example, Rivers and Koedinger used tree edit distances to compute similarities between syntax trees of Python programs to identify adjacent states [16]. Gross and colleagues similarly applied edit distances on syntax trees to infer clusters of computer programs and select the most similar sample solution for feedback [2, 3]. Finally, Paasen, Mokbel and Hammer have identified the underlying algorithm of sorting programs using machine learning techniques based on alignment distances and adapted the parameters of those alignment distances to yield better classification results [11, 13]. Note that all these approaches rely on alignment distances on program syntax, not on execution traces. Striewe and Goedicke applied sequence alignment on execution traces, but did not apply the alignment distances for further datamining purposes [18].

2.4 Classification of Computer Programs

Recently, the value of classification methods for feedback provision in intelligent tutoring systems for computer programming has been recognized. Such machine learning methods enable tutoring systems to infer e.g. the underlying programming strategy of a learner with explicit human labelling only for a small example set [13]. Piech and colleagues report multiplication factors of up to 214, that is, a human tutors annotation for one program permits inference of said annotation for up to 214 other programs [14]. Of course, such approaches rely on a representation of computer programs in a format that can be fed into machine learning methods, such as pairwise similarities and dissimilarities [9, 13] or an explicit vectorial embedding [14]. In this contribution, we employ a classification paradigm to distinguish between different algorithmic approaches, as well as between erroneous and correct solutions.

3. REPRESENTING COMPUTER PROGRAMS VIA EXECUTION TRACES

In general, execution trace recordings can be defined as the “detection and storage of relevant events during run-time, for later off-line analysis” [6]. More specifically, we consider executions of statements in the program as relevant events, which we characterize by the value of variables of interest after the statement has been executed. This is equivalent to a step-wise execution of the program in a debugger, where we record the state of an interesting variables in each step [17]. As an example, consider traces in Table 1 for the programs in Figure 1.

Only modest technical requirements have to be fulfilled to apply a trace representation. 1.) The programming language has to offer a debugging environment which permits monitoring of a program’s execution; 2.) a valid and non-trivial example input for the task has to be available; and 3.) the student’s program has to compile and execute without errors on the example input [17]. Thus, the trace representation is more demanding compared to the very flexible syntactic representation of computer programs, but has less prerequisites compared to extensive knowledge engineering. In that sense, the trace representation can be seen as a “middle road” between entirely data-driven approaches and systems based on expert knowledge.

4. COMPARING EXECUTION TRACES

If a student’s program is analyzed via test cases, the output is compared with the pre-defined reference value via a simple equality test. However, such a strict equality test is not a viable option for the comparison of execution traces. For example, the traces on the left and the middle in Table 1 are not equal. But they are more similar to each other than to an erroneous program that does not sort the input array at all. Therefore, we require a more flexible measure of similarity or dissimilarity between traces [9].

Similarities and dissimilarities on sequential data can be obtained via alignment distances or edit distances. The overarching scheme is to extend both input sequences such that there length becomes equal and similar elements of both sequences become aligned. The alignment distance is then defined as the summed cost over all aligned elements [13]. The choice of alignment algorithm depends on the extensions of input sequences that should be permitted. In case of execution traces we intend to abstract from sequence elements that leave the relevant variables unchanged. As an example, consider lines two and three of the program in Figure 1.

These two lines could be removed from the program without changing its function, if all expressions of r and l are replaced by their value in the rest of the program. A classic edit distance scheme would punish this with a higher dissimilarity between the shorter and the longer version of the program. Instead, we propose that the same state of the relevant variables may be copied without cost. This corresponds to the dynamic time warping dissimilarity DTW for speech processing, first introduced by Vintsyuk [20]. Given two traces \( x = (x_1, \ldots, x_M) \) and \( y = (y_1, \ldots, y_N) \) as well as a dissimilarity measure \( d(x_i, y_j) \) between the variable states...
x_i and y_j, it is defined recursively as:

\[
D_{DTW}(x_1, \ldots, x_i, y_1, \ldots, y_j) := d(x_i, y_j) + \min \begin{cases} \\
D_{DTW}(x_1, \ldots, x_{i-1}, y_1, \ldots, y_j) \\
D_{DTW}(x_1, \ldots, x_i, y_1, \ldots, y_{j-1}) \\
D_{DTW}(x_1, \ldots, x_i, y_1, \ldots, y_j)
\end{cases} \\
D_{DTW}(x_1, y_1) := d(x_1, y_1)
\] (2)

This can be calculated efficiently in \(O(M \cdot N)\) via dynamic programming \(D_{DTW}\) is tabulated for all prefixes of \(x\) and \(y\).

An illustration of the dynamic time warping dissimilarity between two example traces is shown in Figure 2. The first three array states of the left trace are just repetitions and thus are aligned with the first array state of the right trace. This occurs again for the fourth to sixth array state of the left trace. Only afterwards the array states differ and lead to a non-zero dissimilarity between both traces. Note that the explicit alignment of array states between two compared traces in dynamic time warping can be retrieved efficiently via backtracing in linear time.

As other edit distances, the dynamic time warping algorithm crucially relies on a dissimilarity measure between variable states. If prior knowledge regarding the interesting variables is available, defining such a measure becomes fairly straightforward (e.g. a Hamming-distance on arrays, just counting the number of unequal entries). In absence of such prior knowledge, defining a dissimilarity on variable states becomes a challenge in itself. One has to infer a semantic matching between the variables in both programs, determine their relevance (as some variables might be less central to the semantic function than others) and then compute the relevance-weighted distance between all matched variables. As a first step in this direction, we propose a simple summary scheme. We build a histogram \(H_{x_t}\) in each state \(x_t\) that counts the number of variables of each type \(t \in \mathcal{T}\), and compare these histograms with a normalized L1 distance:

\[
d(x_i, y_j) := \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{|H_{x_t}(t) - H_{y_t}(t)|}{|H_{x_t}(t)| + |H_{y_t}(t)|}
\] (3)

Note that we consider only types \(t\) which occur in both programs at least once.

5. EXPERIMENTS
Our experimental evaluation concerns three key challenges for data-driven intelligent tutoring systems: 1.) Identifying the underlying algorithmic approach, 2.) Identifying erroneous programs, and 3.) detecting the location of an error, once a program is identified as erroneous. We compare the performance on these tasks between the trace representation (with dynamic time warping as dissimilarity measure) and the state-of-the-art in terms of syntax representation: syntax-trees with learned edit distance parameters via machine learning techniques [13]. As implementation of the alignment techniques we applied the TCS Alignment Toolbox [12].

5.1 Datasets
For our evaluation, we use two benchmark datasets. The palindrome data set consists of 48 (correct) programs deciding whether all words in an input sentence are palindromic, using one of eight different programming strategies [9]. We used the histogram-approach to define a dissimilarity between variable states and generated traces using the input sentence “OTTO ANNA MOPS”. As this data set does not contain erroneous programs, we only used it for the first experiment.

The second dataset is an extended version of the sorting dataset from [11]. It consists of 126 (correct) sorting programs collected from various web sources, each implementing one of six sorting algorithms (35 BubbleSorts, 29 InsertionSorts, 15 MergeSorts, 17 QuickSorts, 20 SelectionSorts and 10 ShellSorts). For each of the programs we created an erroneous counterpart, with one or more semantic errors, that is, errors that are neither detected by the compiler nor do they lead to a program crash (e.g. due to an index being out of bounds). Thereby, we focused on errors that are non-trivial to detect for technical systems. As a dissimilarity between variable states we employed a Hamming distance on the array to be sorted. As input we generated a uniform random array of 10 integers in the range [0, 99].

Both datasets are available online at http://doi.org/10.4119/unibi/2900666 and http://doi.org/10.4119/unibi/2900684 respectively.

5.2 Classifying Programming Strategies
Our first experiment concerns the identification of the underlying sorting algorithm. We assume that a human expert has already labelled some example programs and want to infer the correct label for some new, unlabelled program. We evaluate the classification accuracy of an 1-nearest neighbor classifier for the syntactic as well as the trace-based representation in a crossvalidation with 6 folds (for the palindrome dataset) and 10 folds (for the sorting dataset) respectively.

The results are shown in Table 2. For the palindrome dataset, the accuracy for the trace representation is more than 10% higher compared to the syntactic representation. Yet, likely due to the small sample size, this difference is not significant (Wilcoxon rank-sum test). In case of the sorting data set,
Figure 3: The sorting dataset embedded in 2 dimensions via t-stochastic neighborhood embedding (t-SNE) [19]. The sorting algorithms are indicated by color. On the left side, the embedding is shown for adapted, syntactic edit distances [13]. On the right side, we show the embedding for dynamic time warping dissimilarities on traces.

<table>
<thead>
<tr>
<th></th>
<th>palindromes</th>
<th></th>
<th>sorting</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>method</td>
<td>acc.</td>
<td>std. dev.</td>
<td>acc.</td>
</tr>
<tr>
<td>syntax</td>
<td>0.875</td>
<td>0.158</td>
<td>0.812</td>
<td>0.068</td>
</tr>
<tr>
<td>traces</td>
<td>0.979</td>
<td>0.051</td>
<td>0.954</td>
<td>0.040</td>
</tr>
</tbody>
</table>

Table 2: The mean classification accuracy and its standard deviation of a 1-nearest neighbor classifier distinguishing six different sorting algorithms. Mean and standard deviation are calculated across 6 (for palindromes) and 10 (for sorting) crossvalidation trials.

we gain an increase in accuracy of more than 14%, which is highly significant ($p < 0.01$, Wilcoxon rank-sum test). This is also reflected in the corresponding dissimilarities. In Figure 3 we show 2-dimensional embeddings of the sorting dataset according to syntax-based (left) and trace-based (right) dissimilarities. The trace representation yields more compact clusters corresponding to the correct class label, thereby making classification easier. Interestingly, closer inspection of the misclassified data points for the trace representation revealed that the 1-nearest neighbor classifier correctly identified a BubbleSort implementation the programmers had wrongly labelled as an InsertionSort.

In order to apply a classification algorithm in praxis, labelled data is required. To reduce human work, one would like to reduce the amount of labelled data necessary. We tested the required amount of labelled data experimentally, by reducing the number of labelled data points (and increasing the number of unlabelled points). The results are displayed in Figure 4. For the palindromes data set, only two data points per class are sufficient to achieve good performance. For the sorting data set, about 40 labelled programs suffice to achieve a classification accuracy of 90% using the trace representation, while the classification accuracy for the syntactic representation saturates at 80% for about 60 programs.

5.3 Classifying Erroneous Programs

We phrase the identification of erroneous programs as a classification task as well: We assume that a human expert

Figure 4: The classification accuracy on the strategy classification task using the syntactic as well as the trace-based data representation if the number of available labelled data points is reduced and the number of unlabelled points is increased. The upper plot displays the result for the palindromes dataset, the lower plot for the sorting dataset. The error-bars mark the standard deviation across 6 and 10 crossvalidation trials respectively.
Table 3: The mean classification accuracy and its standard deviation of a 1-nearest neighbor classifier distinguishing erroneous from correct sorting programs. Mean and standard deviation are calculated across 20 crossvalidation trials.

<table>
<thead>
<tr>
<th>method</th>
<th>Accuracy</th>
<th>std. dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>syntax</td>
<td>0.211</td>
<td>0.107</td>
</tr>
<tr>
<td>traces</td>
<td>0.861</td>
<td>0.086</td>
</tr>
</tbody>
</table>

5.4 Detecting Error Locations

As a final challenge, we try to locate the errors within the erroneous programs. More precisely, the challenge is to identify a set of lines of code in an erroneous program, such that all errors are included, but few other lines are included. Such a set of lines can then be utilized in an intelligent tutoring system. The identified lines can be highlighted such that the student is able to find the error in her program. We apply two strategies based on alignment algorithms, one on the syntactic representation and one on the trace representation.

Syntax-Based Error Detection. We select the nearest correct neighbor and retrieve a syntactic alignment of the erroneous program and the correct program via backtracing. Thereby we obtain the contribution of each line of code in the erroneous program to the overall alignment distance. In order to identify contributing neighbors as well, we apply Gaussian blur to this distribution and then select the line of code with the highest contribution as well as its neighbors, if their contribution is sufficiently high (at least half as high compared to the maximum).

Trace-Based Error Detection. Our trace-based strategy is similar to the syntax-based one. We again select the nearest correct neighbor and retrieve a trace alignment of the erroneous program and the correct program via backtracing. However, we can apply additional domain knowledge. We assume that an erroneous program has the wrong output given the input. The output of the program includes the value of the relevant variables at the end of the trace. Therefore, we can start from the end of the trace alignment and work back until the state of the relevant variables is equal to the state in the correct program. This is the point where the error in the program influences the programs execution negatively. However, it is not sufficient to highlight this particular line of code, because the actual error might be earlier in the code (e.g. a wrongly set index). Therefore, we select not only this line, but the most frequently executed five lines of code until the last change of the relevant variables.

Further, we included three trivial baseline strategies for comparison: 1.) Selecting a line of code at random, 2.) selecting a line of code at random according to its distribution in the trace, and 3.) selecting all lines in the program that occurred in the trace.

We evaluated all five strategies in a 20 fold crossvalidation. For each erroneous program, we excluded the correct counterpart from the available neighbors in order to make the scenario more realistic.

The results are shown in Table 4. We report the classic pattern recognition measures precision (how many of the selected lines of code contain an error?), recall (how many of the erroneous lines of code have been selected?) and F1-score (harmonic mean of precision and recall). In terms of F1-score, the trace-based error detection method clearly outperforms the syntax-based one ($p < 10^{-4}$, Wilcoxon rank-sum test). Further, as expected, both random baseline meth-
ods seldomly select an erroneous line, thereby limiting the recall. However, selecting all lines of code occurring in a trace provides a strong baseline to compete with \( F_1 = 0.186 \). Still, the trace-based error location method performs significantly better \( (p < 0.01, \text{Wilcoxon rank-sum test}) \).

6. DISCUSSION

In this contribution we introduced an alternative representation of computer programs for classification and error detection in intelligent tutoring systems (ITSs), namely execution traces. On two example data sets we have demonstrated that this representation can improve upon state-of-the-art syntax-based representation in terms of strategy classification, error classification and error detection. In a full-blown ITS for computer programming, the trace representation can thus be applied to help students in solving programming tasks. As soon as a student has managed to reach a working state (without syntax errors and program crashes) we can generate a trace and compare it with the traces of different programs. The resulting (dis-)similarity measure can be used to identify possible partners for peer-review and peer-tutoring by matching students that apply the same approach in their solution attempt. Further, the trace representation can be applied to identify erroneous programs, enabling an ITS to detect whether a student has finished a task or still needs to continue. Further, as not only the end result is checked but the whole execution, the trace representation can be utilized for detecting unusual or deceptive solutions that are geared towards the test cases without actually implementing the desired function. Finally, if an error is still present in a student’s program but the error is not obvious, the trace representation may help to identify and highlight the location of the error in the program code, thereby providing scaffolding to students that get stuck in searching for their error.

Overall, the trace representation appears to be highly useful for data-driven ITSs on computer programming. However, important challenges remain. If no a priori knowledge regarding the relevant variables in the program is available, computing a dissimilarity on variable states is not trivial. We have suggested a first attempt using a histogram of variable types. This representation, however, disregards the content of variables and thus is likely not sufficiently powerful in many applications where differences in variable values are important markers of program semantics. A solution might be to match variables probabilistically according to the alignment distance a certain matching produces. This is an interesting direction to pursue in further research.

Finally, we note that the trace representation does not have to be the sole source of information for an ITS. A syntactic representation is necessary when a program does not yet compile or crashes and wherever the high level of abstraction applied by a program trace is not helpful (e.g. when teaching certain syntactic constructs). Fusing the strengths of both representations is likely to lead to the best learning outcomes for students.

Table 4: The mean classification accuracy and its standard deviation of a 1-nearest neighbor classifier distinguishing erroneous from correct sorting programs. Mean and standard deviation are calculated across 20 crossvalidation trials.

<table>
<thead>
<tr>
<th>method</th>
<th>precision</th>
<th>std. dev.</th>
<th>recall</th>
<th>std. dev.</th>
<th>F1 score</th>
<th>std. dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>traces</td>
<td>0.183</td>
<td>0.071</td>
<td>0.520</td>
<td>0.211</td>
<td>0.269</td>
<td>0.104</td>
</tr>
<tr>
<td>syntax</td>
<td>0.103</td>
<td>0.086</td>
<td>0.134</td>
<td>0.100</td>
<td>0.115</td>
<td>0.091</td>
</tr>
<tr>
<td>traces_random</td>
<td>0.157</td>
<td>0.122</td>
<td>0.119</td>
<td>0.098</td>
<td>0.134</td>
<td>0.107</td>
</tr>
<tr>
<td>syntax_random</td>
<td>0.121</td>
<td>0.116</td>
<td>0.095</td>
<td>0.095</td>
<td>0.105</td>
<td>0.103</td>
</tr>
<tr>
<td>traces_all</td>
<td>0.103</td>
<td>0.022</td>
<td>0.976</td>
<td>0.050</td>
<td>0.186</td>
<td>0.037</td>
</tr>
</tbody>
</table>

Figure 5: The sorting dataset including erroneous solutions embedded in 2 dimensions via t-stochastic neighborhood embedding (t-SNE) [19]. The correctness of each program is indicated by color. On the left side, the embedding is shown for adapted, syntactic edit distances [13]. On the right side, we show the embedding for dynamic time warping dissimilarities on traces.
7. ACKNOWLEDGMENTS
Funding by the DFG under grant number HA 2719/6-2 and the CITEC center of excellence (EXC 277) is gratefully acknowledged.

8. REFERENCES
Generating Data-driven Hints for Open-ended Programming

Thomas W. Price
North Carolina State Univ.
890 Oval Drive
Raleigh, NC 27695
twprice@ncsu.edu

Yihuan Dong
North Carolina State Univ.
890 Oval Drive
Raleigh, NC 27695
ydong2@ncsu.edu

Tiffany Barnes
North Carolina State Univ.
890 Oval Drive
Raleigh, NC 27695
tmbarnes@ncsu.edu

ABSTRACT
Intelligent Tutoring Systems (ITSs) have shown success in the domain of programming, in part by providing customized hints and feedback to students. However, many popular novice programming environments still lack these intelligent features. This is due in part to their use of open-ended programming assignments, which are difficult to support with existing hint generation techniques. In this paper, we present a new data-driven algorithm, based on the Hint Factory, to generate hints for these open-ended assignments. We evaluate our algorithm on historical student data and show that it can provide hints that successfully lead students to solutions from any state, help students achieve assignment objectives, and align with the student’s future solution.

1. INTRODUCTION AND BACKGROUND
Intelligent Tutoring Systems (ITS) have shown much promise in the domain of computer programming [3, 14, 16, 22], with studies arguing that using an ITS perform as much as two standard deviations higher than those who receive conventional instruction [3]. A key feature of any ITS is the ability to give students context-sensitive feedback during problem solving, often in the form of hints. In the domain of programming, this feedback has been shown to improve students' performance, both inside the tutor and on subsequent assessment [4].

Despite positive empirical evaluations, these specialized ITSs are not generally used in introductory programming classes. In particular, new introductory Computer Science (CS) curricula, such as CS Principles\(^1\) and Exploring CS\(^2\) are turning to programming environments designed specifically for novices, such as Scratch [19], Snap [8] and Alice [5], which engage students in creating open-ended projects, such as games, stories and simulations [25]. These environments have features specifically designed for novices, such as drag-and-drop, block-based interfaces that improve student performance by minimizing the challenges of syntax [18]. They offer improved outcomes over traditional instruction, such as increased retention [11] and improved test scores [5].

Unfortunately, aside from some preliminary research [2], little effort has been made to bring the intelligent features of ITSs to these novice programming environments. This is due in part to the large investment of time required by domain experts to create these systems, which has been estimated as high as 300 hours to create one hour of intelligent content [12]. Further, the use of open-ended programming assignments, which makes these environments so appealing to students and teachers, also serves as a major barrier to providing intelligent, adaptive feedback. These assignments often have multiple, loosely ordered objectives, which cannot be assessed automatically, making it difficult to apply automatic hint generation techniques that rely on test cases (e.g. [15, 22, 23]).

Data-driven tutors have the potential to overcome these barriers. The Hint Factory is an algorithm that has been used to generate data-driven hints from historical student data, originally in the domain of logic proofs [1]. The Hint Factory is like a recommender system that uses student data as a basis for automatic hint generation, making it easy to scale up without additional expert involvement. The Hint Factory has been successfully adapted to the domain of programming in a variety of ways [14, 9, 22]. However, data-driven hints have not been evaluated on open-ended assignments in novice programming environments, and may not be well equipped to handle them [17].

In this paper we present an extension of the Hint Factory specifically designed to provide hints to students working on open-ended programming assignments. The algorithm is fully data-driven, requiring no reference solution or test cases, and presents hints that represent real student actions. It is designed to be programming language and system agnostic, with the intention of making it applicable to a variety of novice programming environments. We evaluate this algorithm on historical student data from an open-ended assignment in a novice programming environment, and show that it is capable of providing hints that successfully lead students to solutions from any state, help students achieve assignment objectives, and align with the student’s future solution.

1.1 The Hint Factory
The Hint Factory [24] is an algorithm for generating next-step hints for students working on multi-step problems. It operates on a data-structure called an interaction network [6], which is built from log data of the interactions between students and a learning environment for a given problem. The interaction network is a directed graph, where each vertex represents a state of the problem. In programming a state corresponds to a snapshot of the student’s current work (code). States are connected by edges, which represent student actions, such as adding, editing or deleting code, which transform one state into another. Each student attempt is traced from a start state to its final state and is added to the interaction network. If this final state is a correct solution, we label it as a goal state. By combining all students’ attempts into a single network and weighting edges

\(^1\)www.esprinciples.org
\(^2\)www.exploringcs.org
with the number of attempts that passed through them, the interaction network forms a compact representation of student problem solving strategies for a given problem.

The Hint Factory uses the interaction network for a given problem to generate hints for new students working on that problem. When a student requests a hint, the algorithm matches that student to an existing state in the network and then calculates the best path from that state to a goal state. The Hint Factory uses a Markov Decision Process (MDP) to calculate this solution path [1], but other techniques can also be used, which are more effective in some contexts [16]. Once a solution path is calculated, it is typically used to provide a next-step hint, which points the student towards the next state in the solution path. The exact method of suggesting this state as a hint is system-dependent.

1.2 Hint Generation in Programming

The domain of computer programming presents a serious challenge for automatic hint generation, especially for data-driven systems. Even for simple programming problems, the space of possible solutions is quite large, often infinite, and there may be little overlap among student solutions [17, 20]. Many automated hint generation algorithms search through this space, attempting to transform a student’s current program into a solution state using some sort of program generation or synthesis [10, 15, 22, 23, 26]. These techniques require an expert-supplied reference solution and/or set of test cases to ensure that generated programs are correct. To facilitate this transformation, algorithms often represent a student’s program using an Abstract Syntax Tree (AST), a directed, rooted tree where each node represents a program element, such as a function call, control structure or variable, and the hierarchy of the tree represents how these elements are nested together.

Zimmerman and Rupakheti [26] use a pq-Gram tree edit distance algorithm to match a student’s program to its closest counterpart in a database of target solutions, as well as to identify the set of insertions, deletions and relabelings that will directly transform the student’s AST into this solution. Rather relying on a fixed set of solutions, Singh et al. [23] use program synthesis to generate a new solution from the student’s current program. They do so using an expert-provided Error Model, which defines a set of potential transformations to a student’s code for a given problem. Other techniques are data-driven like the Hint Factory, using previous student solutions to provide hints. Perelman et al. [15] also employ program synthesis to search for a solution program, using a Domain-Specific language (DSL) to define possible program transformations; however, they show that this DSL can be automatically generated from previous student solutions. Our approach also works to transform a student’s program into a solution, but rather than using an automated technique like program synthesis, we use edits from actual students. Lazar and Bratko [10] employ a similar approach, applying single-line edits observed in previous student work to transform a student’s program into a solution; however, their technique requires a set of test cases to evaluate generated programs, and ours does not.

The Hint Factory has also been adapted to the domain of programming, with modifications to address the large state space and lack of overlap among student solutions. Rivers and Koedinger [22] extend the Hint Factory using a strategy called path construction to generate a path from a student’s current state to a previously observed goal state, rather than relying on observed student paths. They compute a change vector of all edits needed to transform the student’s current state into the goal state and test to see if any closer solutions are discovered along the way. Pedlycord III et al. [14] applied the Hint Factory to a programming game called BOTS, but rather than representing a student’s state using an AST (a codestate), they used the state of the game world after running the student’s program (a worldstate). The authors found considerably more overlap among worldstates than codestates, allowing more hints to be generated; however, these hints may be more challenging to apply. Fossati et al. [7] used a similar approach to the Hint Factory to generate both reactive and proactive data-driven feedback in the iList linked list tutor. They found that with this feedback, iList produced equivalent learning gains to a human tutor.

Most methods for hint generation benefit from overlap among student programs. This overlap can be increased through canonicalization, which standardizes the syntax of programs, while maintaining their semantic meaning. For example, the expression \( a > b \) can be rewritten \( b < a \) without changing its meaning. Rivers and Koedinger [20] present a comprehensive technique for canonicalization, which standardizes programs in a variety of ways, such as normalizing arithmetic and boolean operators, removing unreachable and unused code and inlining helper functions. Jin et al. [9] take a different approach, representing a student’s program as a Linkage Graph, where each vertex is a code statement, and each directed edge represents an ordering dependency. This removes some semantically unimportant ordering information from the program, allowing for more overlap.

2. THE CTD ALGORITHM

In this section we present the Contextual Tree Decomposition (CTD) algorithm for hint generation, our extension of the Hint Factory to the domain of open-ended programming problems. Existing hint generation techniques are effective on traditional programming assignments with single objectives that are easily assessed with test cases. Open-ended assignments, by contrast, may have multiple, loosely ordered objectives that do not lend themselves to automated assessment, as they often deal with user interaction or graphical output. As such, we cannot rely on the program generation techniques discussed in Section 1.2 to create hints. Instead, we take a fully data-driven approach, using student data, rather than automated search, to construct a path to a goal state. Not only does this approach make hint generation feasible for open-ended assignments, it also has the advantage of presenting hints that correspond to real student actions, which should be understandable to other students.

2.1 An Example Assignment

To illustrate the CTD algorithm, we will use an assignment called the “Guessing Game” as a running example throughout this section. In the Guessing Game, students are asked to create a program that stores a random number and then repeatedly asks the player to guess it until they are correct, informing them if they have guessed too high or too low. To begin, the game should welcome the player and greet them by name. The game should then ask the player to guess the number until they are correct, informing them of their progress. After the game is over, the player should see a summary of their results. The assignment requires the use of loops, conditionals, variables and various arithmetic operators. A common implementation of the Guessing Game is presented in Figure 1.

Note that this is one of many possible solutions to the problem. For example, we could use three if statements, rather than an if/else block. Now consider a student, Alice, working on
GuessingGame:
Say("Welcome to the Guessing Game!")
answer ← Ask("What is your name?")
Say(Join("Hello ", answer))
number ← Random(1, 10)
doUntil(answer == number):
   answer ← Ask("Guess a number")
   if (answer == number):
      Say("Correct!")
   else:
      if (answer > number):
         Say("Too high!")
      if (answer < number):
         Say("Too low!")

Figure 1: An example solution to the Guessing Game assignment.

GuessingGame:
number ← 8
Say("Welcome!")
answer ← Ask("Who’s playing?")
Say(Join("Hi ", answer))
doUntil(answer == Random(1, 10)):
   answer ← Ask("Guess a number")

Figure 2: An example of a partial, flawed solution attempt from a student, Alice.

the Guessing Game with code presented in Figure 2. Alice has added the first few lines of code in a different (but correct) order; however, she does not understand how to store and use the random number for the guessing game. A hint could demonstrate the correct behavior for her.

2.2 Generating Hints

In the CTD algorithm, as in previous work, we represent a student’s state using an AST. Borrowing from Rivers and Koedinger’s work [20], we also use basic canonicalization to increase overlap among ASTs. In our ASTs, we use a single label for all variables (var) and for all literals (literal). The arguments of commutative operators (e.g., ==, +, *) are given a fixed ordering, and we rewrite any greater than expression \( x > y \) as a less than expression \( y < x \). A canonicalized AST for the code presented in Figure 1 is shown in Figure 3.

Most data-driven hint generation algorithms attempt to answer the question, “Given a student’s current state, what should their next state be?” Rather than trying to answer this question for a student’s entire program, we try to answer it for the children of each node of a student’s AST. For example, if Alice were to request a hint, we might tell her to assign a different value to number, compare different values using == or add code to the body of doUntil. By breaking the student’s program down into a set of smaller pieces, we can more easily match it to the programs of previous students, as suggested in previous work [10, 21].

To generate hints from student data, we build a set of contextual interaction networks (CINs), which each model how students edit a subsection of the program over time. We build one CIN for each unique root path observed in all students’ ASTs (including ASTs from intermediate code snapshots). A root path (RP) for a node in an AST is the path from the root node to n. Figure 3 highlights an example RP for the (==) node: (script, doUntil, ==). Some nodes have the same root path, such as the two (Say) nodes, which have the RP (script, Say). Each RP corresponds to a unique CIN, denoted CIN(r), which functions just as the interaction networks described in Section 1.1. However, CIN(r) only models changes to the immediate children of the last node in r. For example, CIN((script, doUntil, ==)), shown in Figure 4, models changes to the children (operands) of the (==) node. Each state in CIN(r) is a list of the children of the last node in r, and each edge represents an edit to those children. Figure 3 highlights C_g, the list of children of the (==) node, which corresponds to a state in the CIN shown in Figure 4. Because the AST shown in Figure 3 is a correct solution, C_g is a goal state in CIN((script, doUntil, ==)). Given that Alice’s current state in this CIN is [var, Random], to get to the goal state C_g we would recommend that she delete her (Random) node and then replace it with a (var) node.

The procedure for building the CINs from previous data is shown in Algorithm 1. We represent a student’s work as a sequence of ASTs, \( T \), where each tree \( t_i \) in the sequence is a snapshot of the student’s work at time \( i \), and the last tree represents the submitted solution attempt. For each sequential pair of trees, \( t_i \) and \( t_{i+1} \), we find all pairs of AST nodes \((n_i, n_{i+1})\) that represent

Figure 3: A partial AST for the code shown in Figure 1. A root path \( r \) is outlined in bold blue, with its current state \( (C_g) \) in dashed green.

Figure 4: The contextual interaction network CIN((script, doUntil, ==)) with goal state \( C_g \). Edge thickness represents transition frequencies.
the same code element in both trees, and therefore have the same
RP \ r. We examine the lists of child nodes \ C_i \text{ of } n_i \text{ and } C_{i+1} \text{ of } n_{i+1} \text{ in their respective ASTs}. If \ C_i \text{ and } C_{i+1} \text{ are different, we}
add the states \ C_i \text{ and } C_{i+1} \text{ to CIN}(r) \text{ (} \text{if they do not already exist)} \text{ and add an edge from } C_i \text{ to } C_{i+1}. \text{ This edge represents}
how the student has edited the code in this part of the AST
from time \ t_i \text{ to } \ t_{i+1}. \text{ Algorithm 1 runs in } O(|T||S_n|^2) \text{ time for}
a given student, where } |T| \text{ is the number of ASTs recorded
for that student and } |S_n| \text{ is size of the largest recorded AST.}

Algorithm 1: Add a Student to the CINs

Require: A sequence of student ASTs \ T
Ensure: \text{Student data has been added to relevant CINs}
\begin{algorithmic}
\For {all } t_i, t_{i+1} \in T \Do
\For {all } (n_i, n_{i+1}) \in MATCHINGNODES(t_i, t_{i+1}) \Do
\State \ r \gets \text{ROOTPATH}(n_i)
\State C_i \gets \text{CHILDREN}(n_i)
\State C_{i+1} \gets \text{CHILDREN}(n_{i+1})
\If {C_i \neq C_{i+1}}
\State \text{ADDEDGE(CIN}(r), C_i, C_{i+1})
\EndIf
\EndFor
\EndFor
\end{algorithmic}

Once we have added student data to the CINs, we can generate
hints for new students, as shown in Algorithm 2. Because we
now have many CINs, rather than a single interaction network,
we also generate a set of hints. For each node \ n \text{ in a student’s}
current AST, we calculate its root path \ r \text{ and find CIN}(r).
The student’s current state in CIN(r) is \ C_0, \text{ the list of children of}
\ n. \text{ We then use the Hint Factory algorithm [1] to generate a}
hint using the interaction network CIN(r) \text{ and the student’s}
current state in the network } C_0. \text{ This hint will recommend a}
new set of children \ C_1 \text{ for } n, \text{ which we can then display as a}
suggestion to the student. Note that if } C_0 \text{ is already a goal
state the Hint Factory will recommend that the student stay
in that state, in which case } C_0 = C_1 \text{ and we present no hint for
} n. \text{ Algorithm 2 runs in } O(|T|^2 + |S_n|^2) \text{ time } ^3, \text{ where } |T| \text{ is the size of the student’s AST and } |S_n| \text{ is the number of states in
the largest CIN}(r). \text{ In practice, } |S_n| \text{ remains small, as a given
CIN models changes to only a small part of a student’s code.}

Algorithm 2: Get Hints

Require: The student’s current AST \ t
Ensure: \text{H is a set of node-hint pairs}
\begin{algorithmic}
\State \ H \gets \{\}
\For {all } n \in NODES(t) \Do
\State \ r \gets \text{ROOTPATH}(n)
\State C_0 \gets \text{CHILDREN}(n)
\State C_1 \gets \text{HINTFACTORYHINT(CIN}(r), C_0)
\State \ H \gets H \cup \{(n, C_1)\}
\EndFor
\end{algorithmic}

A classic challenge for the Hint Factory is how to provide hints
to states with no exact matches in the interaction network.
CINs break a program down into smaller parts to provide more
opportunities for matches, but this does not guarantee a match.
If no exact match is found for a state \ C_0\text{, we find the closest
state to } C_0 \text{ in the CIN and use it as a next-step hint. Because}

\text{CIN states are lists of children, we can use a simple edit dis-
tance to determine the closest state. If the distance between
the current state and its closest pair in the CIN is beyond a
certain threshold (e.g. 3 edits), we assume the student is doing
something unknown, and we do not provide a hint for that state.}

2.3 Goal States

In order to run on an interaction network, the Hint Factory
requires a reward function \ R(s), \text{ which is used by the MDP
to assign a reward to each state in the network [1]. Traditionally,
this value has been some large number (e.g. 100) for goal states
and 0 otherwise. However, in many open-ended programming
problems, we cannot automatically determine whether or not a
given program state satisfies the goal of the assignment. A simple
solution is to assign a reward value to each state proportional
to the number of students who submitted a program in that
state. We accomplish a similar effect with CINs by finding each
node \ n \text{ and corresponding RP } \ r \text{ in each student’s submitted
AST and marking the list of children of } n \text{ as a goal in CIN}(r).

One challenge with CINs is that two different parts of a program
can correspond to the same CIN. For example, recall that the
two \text{Say statements in Figure 3 have the same RP, and thus
the same CIN, but ideally these two nodes should end up in
two different goal states. The first should end up with children
}[\text{literal}], \text{ while the second should have children } \text{[Join]. Both
of these states will be marked as goals in the shared CIN, so how
can the algorithm determine when one goal should be chosen
over the other?}

To address this, each time a node’s children are marked as a goal
state in a CIN, we also store that node’s context. This context
helps identify when a particular goal state might be applicable.
We define a node’s context using two lists, consisting of its left
and right siblings in the AST. For example, in Figure 3, the
first \text{Say node has a context } \{[\text{[}, \text{[}, \text{Say}, \text{[}, \text{[}, \text{do}\text{until]}\}, \text{ while the second node has a context } \{[\text{[}, \text{[}, \text{Say}, \text{[}, \text{[}, \text{do}\text{until]}\}. \text{ Rather than giving goal states a fixed reward value, we
determine this value individually for each hint request. For each
previous student attempt that finished in a given goal state, we}
increase the reward for that state by a value inversely propor-
tional to the distance between the previous attempt’s context and
the current attempt’s context. Again, because the contexts con-
sist of lists, a simple edit distance can serve as a distance metric.

2.4 Smoothing Hints

The Hint Factory is typically used to generate a next-step hint,
which suggests the next state a student should achieve. The
advantage of the Hint Factory is that this action has been done
by a previous student, and is therefore likely to seem reasonable
to the current student. However, sometimes the path that a real
student takes to a solution can be circuitous. Students often add
code that they later delete, or add code in one place and later
move it to another. In these cases we use the entire solution
path generated by the MDP, rather than a single state, to make
suggestions that will not be contradicted by future hints. We
call this process “smoothing”, since it will make hints appear
more consistent.

We use Algorithm 3 to generate hints which follow real students’
paths, while avoiding unnecessary or contradictory edits. We
first calculate a full solution path from the student’s current state
to a goal state using the Hint Factory on the CIN, as described
earlier. Recall that each state in this path is a list of child nodes in the AST. We first reorder nodes in the student’s current state to match the goal state ordering. We then insert any new nodes from the next state in the solution path (like set union) and reorder the nodes again to match the goal state. Finally, we remove any nodes that are not in the goal state (like set intersection). If the resulting state is not different than the student’s current state, we repeat the process with the next state in the solution path. Using this “smoothing” process helps us avoid giving hints that add code that will later need to be moved or deleted.

Algorithm 3 Get Smoothed Hint

Require: The MDP of a CIN and student’s state
Ensure: hint is a smoothed hint for the student

\[ \text{path} \leftarrow \text{GetSolutionPath(state, MDP)} \]
\[ \text{goal} \leftarrow \text{Last(path)} \]
\[ \text{hint} \leftarrow \text{state} \]
\[ \text{hint} \leftarrow \text{REORDER(hint, goal)} \]
for all \( s_i \in \text{path} \) do
\[ \text{hint} \leftarrow \text{hint} \cup s_i \]
\[ \text{hint} \leftarrow \text{REORDER(hint, goal)} \]
\[ \text{hint} \leftarrow \text{hint} \cup \text{goal} \]
if \( \text{hint} \neq \text{state} \) then
\[ \text{return} \]
end if
end for

3. METHODS

We evaluated the efficacy of the CTD algorithm using data from real students working on the Guessing Game assignment described in 2.1. Data was collected from an introductory undergraduate computing course for non-CS majors during the Fall of 2015, which had approximately 80 students. The first half of the course focused on learning the Snap programming language through a curriculum based on the Beauty and Joy of Computing (BJC) [8]. Snap is a visual programming environment that allows users to create media-rich, interactive programs by dragging blocks of code together to form scripts. Students worked on the Guessing Game assignment during class for approximately one hour, with a teaching assistant (TA) available to assist them and the ability to discuss the assignment with nearby students. We collected trace log data of all student interactions with the programming environment. After each edit to a student’s program, the complete program state (a snapshot) was recorded. For the “Guessing Game” assignment, we collected 51 attempts, consisting of 8666 total code snapshots.

Each of the final submissions was graded by two independent graders. The graders used a rubric consisting of nine assignment objectives, such as welcoming the player by name, storing a secret number, and repeatedly asking the player for guesses. The graders had an initial agreement of 94.5%, with Cohen’s \( \kappa = 0.544 \), and after clarifying objective criteria and independently re-grading this rose to 98.1%, with Cohen’s \( \kappa = 0.856 \). Any remaining disagreements were discussed to create final grades for each assignment. The students achieved on average 92.8% of objectives, with all students getting at least 4 out of 9. The high grades can be attributed in part to the presence of TAs, who helped struggling students to complete the assignment. Using the same criteria, an automatic grading program was created, which manually checked code structure for objective completion. The automatic grader was tested on the manually graded data, achieving 100% accuracy on 7 of 9 objectives. On each of the remaining two objectives, it incorrectly marked two submissions as failing since they used atypical approaches. Note that this grader was used in our evaluation but not for hint generation.

We generated and evaluated hints for each code snapshot of each student in our dataset (n=8666), giving us a clear view of hint performance across students and time. We evaluated the hints using a number of criteria, detailed in Section 4. Because Snap lends itself to a “tinkering” approach, code snapshots often contain many extra scripts that students keep in their workspace for later use. Since the Guessing Game uses only one script, these extra scripts do not reflect the student’s primary work, and it would not make sense to evaluate hints for them. Therefore, in our analyses we considered only the largest script in a snapshot.

3.1 Hint Policies

To better evaluate the CTD algorithm, we generated hints using four hint policies:

1. **CTD All (CA):** Hints are generated using CTD on all student data (n=856).
2. **CTD Exemplar (CE):** Hints are generated using CTD on data from only exemplar students, whose final submissions achieved all assignment objectives (n=32).
3. **Direct Expert (DE):** Hints modify a student’s program directly towards an expert solution using a single node insertion, deletion or relabeling.
4. **Direct Student (DS):** Hints modify a student’s program directly towards their own submitted solution, using a single node insertion, deletion or relabeling.

The CA and CE policies both use the CTD algorithm, and comparing them allows us to explore the effect of including students with incorrect final solutions on the algorithm’s output. The DE and DS policies both generate hints using a technique outlined by Zimmerman et al. [26], which identifies the node insertions, deletions and relabelings required to transform one AST into another. Each of these modifications is treated as a hint. The DE policy targets a single expert solution, while the DS policy targets the student’s own future final solution, and could not actually be implemented on real-time data. In many ways, the DS policy represents an ideal hint policy for students who achieve a correct final solution (which the majority of our sample did), as it perfectly anticipates their solution strategies.

All policies generate a set of hints, where each hint represents a small modification to a student’s program. When generating hints for a given student using CTD, we did not include that student in the dataset used to build the CINs, similar to leave-one-out cross validation, since that student’s future data could not be used in a real-time setting.

4. EVALUATION AND DISCUSSION

Our evaluation of CTD focused on the following research questions:

RQ1 Can CTD successfully lead students to a solution regardless of their current program?
RQ2 Can CTD hints help students complete objectives?
RQ3 How consistent are CTD hints with student actions?
RQ1 asks whether CTD solves the challenge of generating hints for an open-ended problem where there is little exact overlap among student solution paths. RQ2 investigates whether these hints are good in that they lead students to complete assignment objectives. Lastly, RQ3 asks whether the hints that CTD provides point students in what might be perceived as a reasonable direction, so students will be inclined to use them. Our evaluations for RQ2 and RQ3 compared the CA and CE policies with the baseline DE and DS policies discussed in Section 3.1.

4.1 Providing Hints
RQ1 asks whether CTD can successfully generate hints for solution attempts regardless of how much overlap they have with other attempts in our dataset. Therefore, we first examined how much overlap there was in our dataset. We recorded 8666 snapshots from 51 students; however, many students produced duplicate snapshots, for example by adding and then removing an element of code. If we do not count duplicate snapshots from the same student, we are left with 5103 snapshots. If we also ignore all but the largest script from these snapshots (as is done in our analyses), there are 3181 non-duplicate snapshots. Of these, 2714 (85%) were unique after canonicalization, meaning they showed up in only one student’s data. In addition, 47 of 51 students had unique final solution ASTs. We conclude that the state space is quite sparse, with little overlap among student solution paths.

We evaluated hints from the CA and CE policies to determine if they could get students to a solution despite this sparsity. To align student attempts over time, and to balance our sample evenly across students, we took 50 snapshots from each student, spaced evenly throughout their progression, and called these “slices.” For each student, we generated a hint chain from each of these 50 snapshots to a final solution. A hint chain is the sequence of program states that would result if the students followed sequential “top-level” CTD hints from a given snapshot to program completion. The top-level hint is that which comes from the CIN(r) with the shortest RP r.

Both CA and CE policies were able to generate successful hint chains for every slice, meaning the hint policies always had a hint to provide and there were no hint cycles. Figure 5 shows the average hint chain length for each slice. Both policies showed a steady, near-linear decline in hint chain length over time. This supports the notion that CTD makes good use of the student’s existing work. On average, students took 175.8 steps to complete the assignment, so both policies are more efficient than the student until slice 46/50. As students converge on their own solutions, however, the hints chains become less efficient, as they often lead students to alternative solutions.

To understand the quality of solutions created using hint chains, we evaluated the final solutions generated by the hint chains at each slice using the automatic grader discussed in Section 3. The CA policy solutions received grades averaging from 89.5-93.0% across slices, while the CE policy averaged 98.5-100% across slices. Upon closer inspection, we found that all imperfect CA solutions were identical and satisfied 8 of 9 objectives (88.9%), and all correct CA solutions were identical as well. The one objective missed by the imperfect CA solution was also missed by 12 of 51 students (23.5%), indicating that a frequent enough mistake in student data will be reflected in CTD hints. The CE policy produced 3 unique, correct solutions and 2 unique, incorrect solutions, which both satisfied the same 8 out of 9 objectives. These results suggest that both CTD policies can lead students to high-quality (though sometimes imperfect) final solutions, but exemplar data may be required to generate consistently correct solutions. It is important to note that CTD operates without test cases, and therefore cannot guarantee correctness 100% of the time.

4.2 Objective Satisfaction
To address RQ2, we tested how frequently an available hint would complete an assignment objective before the student did. Figure 6 shows, for each policy in Section 3.1, the percentage of students who had an objective completing hint available for each objective. All hint policies perform fairly well, with at least 45% of students having a completion hint available for objectives 3-9. The CTD policies perform much worse on Objective 2, but otherwise they generally keep pace with the Direct policies. Since these Direct policies offer all edits towards their target solution as hints, they should discover most of the possible completing hints. However, it is important to remember that it is not always possible to complete an objective before a student because hints cannot add more than one node to the AST at a time, while a student’s edit might change many nodes at once by dragging and dropping code.

It is not sufficient for a hint policy to generate good hints; it is equally important that it not generate bad hints. To evaluate this second facet of RQ2, we tested how frequently hints from each policy undid an objective, meaning the objective was satisfied before applying the hint, but it became unsatisfied afterward. Figure 7 compares each policy, showing the percentage of students who received a hint that would undo each objective. Predictably, the DS policy, which anticipates a student’s final solution, performs well across the board. However, the difference between the DE and CE policies is clear. The CE policy stays below 40% on all objectives, and performs as well or better than the DE policy on all but one objective, often by a factor of 2 or more. The CA policy performs slightly worse than the CE policy on most objectives, most notably Objective 1. This can be attributed to the fact that many students did not in fact complete Objective 1, leading the CA policy to suggest removing the code that did so.
We do not make the claim that a good hint always completes an assignment objective, nor that undoing an objective always constitutes a bad hint. Still, these criteria serve as good baseline standards for a hint policy. While all policies are fairly successful at suggesting hints that move students toward completing objectives, the CTD and DS policies avoid undoing objectives much better than the DE policy.

### 4.3 Alignment with Student Actions

RQ3 asks whether or not CTD produces hints which are consistent with a student’s solution path. Ideally, a hint policy should not only provide hints which lead to a good solution; as much as possible, these hints should also make sense to the student receiving them. While the comprehensibility of a hint is impossible to measure without user data, we can approximate this by asking whether or not a hint gets the student closer to their future final solution. Presumably, such hints will seem reasonable to the student, as the student eventually went in that direction on their own.

To answer this question, we examined each hint generated with each policy across all code snapshots and calculated whether or not each hint would get the student closer to their final solution than their original state. We used the Robust Tree Edit Distance algorithm [13] to measure the distance between snapshot ASTs. This metric counts the number of insertions, deletions and relabelings required to transform one AST into another. As a baseline, we also calculated this measure for the student’s own next state, to determine how frequently a student’s actions got them closer to their own final solution state. The results for each policy, averaged over students, are presented in Table 1.

As a baseline, we see that the student’s own next step got closer to their final solution 60.95% of the time. The DS policy, which attempts to directly transform the student’s state into their solution state, achieves only 39.37%, in part because its hints will often delete useful code and later add it again in a better location. However, the DS policy’s performance might be seen as a high target, as it requires future knowledge of the student’s actions. In comparison, we see that the CTD policies both approach the DS policy and far outperform the DE policy. The CE policy gets students closer to their final objective 53.4% as frequently as the student’s own actions and 82.6% as frequently as the DS policy, and the CA policy performs even better.

Post hoc paired t-tests showed that the difference between the CA and DS policies was not significant ($t(50) = 1.63; p = 0.109$), while the difference between the CA and DE policies was significant ($t(50) = 6.96; p < 0.001$). Interestingly, the difference between the CA and CE policies was also significant ($t(50) = 2.67; p = 0.010$), suggesting that restricting data to exemplar students makes CTD hints less reflective of real student behavior. While all policies present some hints that move the student farther away from their final solution, the CTD and DS policies seem to minimize this behavior.

### 5. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a novel algorithm called CTD for generating next-step hints for students working on open-ended programming assignments. Using data from 51 students working on one such assignment, we have shown that the hints generated by the CTD hint policies can get a student to a high-quality solution from all observed states. We have also shown that the hints are capable of helping students accomplish most assignment objectives before they would otherwise do so, without presenting many hints which undo these objectives. Further, CTD produces hints which get students closer to their final solutions relatively frequently. We have also compared the CA policy, which uses all student data, to the CE policy, which uses exemplar data only. While both policies perform well, the CA policy aligns closer with real student actions, while the CE policy produces higher quality final solutions and is less likely to suggest undoing assignment objectives.
Despite these positive initial results, much work remains to be
done to improve CTD. A major limitation of this work is the
reliance on a single assignment for evaluation. Future work
will explore the efficacy of CTD with a variety of assignments.
One challenge that will be presented by larger assignments is
ensuring that the contextual goal matching features discussed
in Section 2.3 work for programs with multiple scripts. Ad-
tionally, while CTD incorporates some of the strategies of
the other hint generation algorithms discussed in Section 1.2,
such as canonicalization, there are others, such as Rivers and
Koedinger’s path construction [22], which could also be incorpo-
rated. Because the CNNs are simply small interaction networks,
any advances to the Hint Factory can also be applied to them.
Lastly, we have already incorporated our hints into the Snap en-
vironment, and future work will investigate how they impact real
students. We will explore the effect of CTD hints on students’
performance on assignments, as well as their learning gains.

6. ACKNOWLEDGMENTS
This material is based upon work supported by the National
Science Foundation under Grant No. 1432156.

7. REFERENCES
Generation for Logic Proof Tutoring Using Historical
Student Data. In Proc. of the 9th Int. Conf. on Intelligent
the 17th Annual ACM ITiCSE Conf., pages 138–143. ACM
Two-Sigma Problem. In Proc. of the 8th Int. Conf. on
Computer-Based Tutoring: Impact on Learning Rate,
Achievement and Attitudes. In Proc. of the SIGCHI
Conference on Human Computer Interaction, pages
[5] W. Dann, D. Coegrove, and D. Slater. Mediated transfer:
Alice 3 to Java. In Proc. of the 43rd Annual ACM
Networks: Generating High Level Hints Based on Network
Community Clustering. In Proc. of the 5th Int. Conf. on
[7] D. Fossati, B. Di Eugenio, S. Ohlsson, C. Brown, and
L. Chen. Data Driven Automatic Feedback Generation in the
iList Intelligent Tutoring System. Technology,
Instruction, Cognition and Learning, 10(1):5–26, 2015.
[8] D. Garcia, B. Harvey, and T. Barnes. The Beauty and Joy of
Representation for Automatic Hint Generation for a
The 11th Int. Conf. on Intelligent Tutoring Systems, pages
1–6, 2012.
The 12th Int. Conf. on Intelligent Tutoring Systems, pages
Effectiveness of a New Instructional Approach. ACM
Analysis of the State of the Art. Int. Journal of Artificial
for the Tree Edit Distance. Proceedings of the VLDB
Hints for Programming Problems Using Intermediate
Output. In Proc. of the 7th Int. Conf. on Educational
Synthesis for Automated Feedback for Introductory
Computer Science Assignments. In Proc. of the Workshop
on Data Mining for Educational Assessment and Feedback,
2014.
Automatically Generating Hints by Inferring Problem
Solving Policies. In Proc. of the 3rd ACM Conf. on
Learning @ Scale, pages 1–10, 2015.
[17] T. W. Price and T. Barnes. An Exploration of
Data-Driven Hint Generation in an Open-Ended
Programming Problem. In Proc. of the Workshop on
Graph-Based Data Mining held at EDM’15, 2015.
[18] T. W. Price and T. Barnes. Comparing Textual and Block
The Int. Computing Education Research Conference,
2015.
K. Brennan, A. Millner, E. Rosenbaum, J. Silver,
Conf. on Intelligent Tutoring Systems, pages 591–593,
2012.
Programming Feedback: A Data-driven Approach. In
Proc. of the First Workshop on AI-supported Education
Generation in Vast Solution Spaces: a Self-Improving
Python Programming Tutor. Int. Journal of Artificial
Intelligence in Education, 2015.
Feedback Generation for Introductory Programming
Assignments. ACM SIGPLAN Notices, 48(6), 2013.
Experimental Evaluation of Automatic Hint Generation
for a Logic Tutor. Int. Journal of Artificial Intelligence
[25] I. Utting, S. Cooper, and M. Kölling. Alice, Greenfoot,
and Scratch – A Discussion. ACM Transactions on
Framework for Recommending Program Elements to
Novices. In Proc. of the 30th Int. Conf. on Automated
How to Model Implicit Knowledge? Similarity Learning Methods to Assess Perceptions of Visual Representations

Martina A. Rau
Educational Psychology
University of Wisconsin—Madison
1025 W. Johnson St
Madison, WI 53706
marau@wisc.edu

Blake Mason
Electrical and Computer Engineering
University of Wisconsin—Madison
1415 Engineering Dr
Madison, WI 53706
bmason3@wisc.edu

Robert Nowak
Electrical and Computer Engineering
University of Wisconsin—Madison
1415 Engineering Dr
Madison, WI 53706
nowak@ece.wisc.edu

ABSTRACT
To succeed in STEM, students need to learn to use visual representations. Most prior research has focused on conceptual knowledge about visual representations that is acquired via verbally mediated forms of learning. However, students also need perceptual fluency: the ability to rapidly and effortlessly translate among representations. Perceptual fluency is acquired via non-verbal, implicit learning processes. A challenge for instructional interventions that focus on implicit learning is to model students’ knowledge acquisition. Because implicit learning is non-verbal, we cannot rely on traditional methods, such as expert interviews or student think-alouds. This paper uses similarity learning, a machine learning method that can assess how people perceive similarity between visual representations. We used this approach to model how undergraduate students perceive similarity between visual representations of chemical molecules. The approach achieved good accuracy in predicting students’ similarity judgments and expands expert predictions of how students might perceive visual representations of molecules.

Keywords
Perceptual knowledge, implicit learning, visual representations, similarity learning methods, chemistry.

1. INTRODUCTION

![Figure 1. Visual representations of chemical molecules: a: Lewis structure; b: ball-and-stick model; c: space-filling model; d: electrostatic potential map (EPM) of water.](image)

Visual representations are ubiquitous instructional tools in science, technology, engineering, and math (STEM) domains [1, 2]. For example, instructors use the visual representations shown in Figure 1 to help students learn about chemical bonding. Yet, to a novice student, these visual representations may not be helpful because the student may not know how to interpret the representations. For instance, does the red color in the ball-and-stick figure (Figure 1-b) mean the same thing as in the electrostatic potential map (EPM; Figure 1-d)? (It does not.)

Instructors often ask students to use visual representations that they have never seen before to make sense of concepts that they have not yet learned about [3, 4], an issue known as the representation dilemma [5]. Hence, to succeed in STEM, students need representational competencies that enable them to use visual representations to make sense of and solve domain-relevant problems [6, 7]. One crucial representational competency is the ability to interpret visual representations; that is, to map visual representations to the abstract concepts they depict [6, 8]. For example, students need to understand how the representations in Figure 1 provide information about the molecule. For the Lewis structure (Figure 1-a), the student may map the unbonded electrons shown as dots to conceptual knowledge about how polarity in chemical molecules and infer that the water molecule has a local negative charge by the Oxygen atom.

Educational technologies are particularly suitable to support representational competencies because they can provide adaptive support while students solve domain-relevant problems [9, 10]. Such adaptive support relies on a cognitive model that infers whether the student has learned target skills based on her/his interactions with the technology. Research shows that adapting instruction to students’ representational competencies can enhance those competencies [11] and learning of domain knowledge [12]. However, educational technologies for representational competencies have two critical limitations. First, they typically focus on one set of representational competencies: students’ conceptual understanding of representations (e.g., the ability to explain how visual features depict concepts). This focus mimics education psychology research’s focus on conceptual learning [6, 13]. Conceptual knowledge is invariably intertwined with a second type of representational competency: perceptual knowledge [14, 15], the ability to rapidly and effortlessly recognize conceptual information based on visual features of the representations. This ability results from implicit forms of learning. For example, expert chemists simply “see” that the molecules depicted in Figure 1 have a local negative charge by the Oxygen atom, without having to make a an effortful conceptual inference.

Second, of the few educational technologies that enhance perceptual fluency, their adaptive capabilities are limited and their perceptual supports rely solely on performance measures (e.g., accuracy, response times) to adapt to students’ representational competencies [15, 16]. They do not use a cognitive model of the latent skills that students acquire through perceptual learning. As a result, they cannot provide specific feedback when students make mistakes. Decades of research showing that cognitive models can dramatically increase the effectiveness of educational technologies [10, 17] suggest that we must address this limitation and create adaptive instruction for perceptual knowledge.

These limitations likely result from cognitive modeling’s traditional focus on explicit, verbally accessible knowledge. To develop cognitive models, researchers analyze how students think about target skills [9, 18]. We typically ask students to verbalize their problem-solving steps [19, 20]. Yet, verbalization is not suitable for assessing perceptual learning processes, which are implicit and not verbally accessible [14, 21]. Therefore, instructional designers have to rely on “educated guesses” as to which visual features students may pay attention to. These educated
guesses are based on the novice-expert literature, which documents the fact that novices tend to rely on surface features; that is, easily perceivable visual cues such as color and shape, to judge the similarity between stimuli items. By contrast, experts rely on visual features that are conceptually relevant and hence make more refined distinctions between visual features. Thus, to create adaptive perceptual supports, we need to develop cognitive models for perceptual learning.

Our research takes a first step towards developing a cognitive model for perceptual learning by assessing students’ perceptual knowledge of a common visual representation in chemistry. In particular, we investigate research question 1: Which visual features do students focus on when presented with visual representations? To address this question, we asked hundreds of students to judge the similarity between visual representations of molecules.

We then used similarity learning—a machine learning method that provides a formal approach to investigating how people perceive similarity among visual stimuli. This method allowed us to estimate latent factors that account for the perceived similarity relationships between representations. Because we can map these latent factors to the visual features in the representations, this approach allows us to investigate which visual features are most salient to students’ perceptions of similarity. Comparing these visual features to “educated guesses” allowed us to test research question 2: Do the visual features we identified as salient via metric learning correspond to visual features that students are expected to attend to based on the expert-novice literature on perceptual learning? In addition, we investigated a methodological research question 3: How many similarity judgments we need to assess students’ perceptual knowledge?

Although we address these questions in the context of a particular domain with a particular visual representation, this paper makes two important broader contributions. First, it provides an empirical validation of the “educated guesses” that developers of perceptual learning technologies typically rely on. Second, it establishes a methodology to assess perceptual knowledge that can serve as a basis for a cognitive model of perceptual learning. These contributions build the foundation for the development of adaptive instructional perceptual knowledge and other implicit knowledge.

2. EXPERIMENT

2.1 Visual Representations of Molecules

For our experiment, we selected visual representations of chemical molecules common in undergraduate instruction. Lewis structure representations are the most commonly used visual representations in undergraduate chemistry textbooks. We reviewed textbooks and online instructional materials and listed the frequency of all occurring molecules using their chemical names (e.g., H2O) and common names (e.g., water). For our experiment, we chose the 50 most common molecules.

First, we created educated guess features (Figure 2, yellow) that correspond to expert assessments of which visual features students may attend to when making similarity judgments. To obtain these educated guesses, we reviewed the literature on chemistry expertise [22, 23] and on perceptual learning [14, 24], and conducted learner-centered interviews with undergraduate and PhD students in chemistry [25]. We identified 6 educated guess features: number of total letters, number of distinct letters, number of total bonds, number of single bonds, number of unbonded electrons, and molecule geometry (linear, planar, tetrahedral).

To investigate which visual features drive students’ similarity judgments, we quantitatively described the visual features of the molecule representations. To this end, we created feature vectors for each of the molecules (see Figure 2, red) that describe which visual features the representation contains (e.g., bond angles, the numbers of specific atoms, or the numbers of different atoms present). The feature vectors of our corpus of molecule representations contained a total of 110 features. The 50 feature vectors collectively form matrix $X = [x_1, x_2, x_3, \ldots, x_{50}]$, where $x_i$ is the feature vector for the $i$th molecule.

We aggregated each element of the feature vectors into molecule vector for individual features (Figure 2, blue). Each molecule vector consisted of 50 values describing how many times the feature occurred in each representation. As molecule vectors make up the rows of our matrix of 110 features by 50 molecules shown in Figure 2, we will refer to the molecule vector for the $j$th feature as $r_j$. Thus, feature vectors provide a numeric description of the visual information present in each representation, whereas molecule vectors provide a numeric description of overall patterns of visual features in the dataset for all representations.

2.2 Similarity Judgment Tasks

Students completed similarity judgment tasks that were presented as triplet comparisons (see Figure 3). Given a representation of a molecule (the “target-molecule”), students were asked to choose molecules”) was most similar to the given one. For each task, the student chose between one of the two choice-molecules that
We delivered the similarity judgment tasks via NEXT; a cloud-based machine learning platform [26]. NEXT allows users to upload their own content and query participants to perform judgment tasks. It uses machine learning algorithms to automate data collection and analyze results. More information about the platform can be found at http://nextml.org. In NEXT, students first received a brief description of the study and then worked through a sequence of 50 similarity judgment tasks. Students were instructed that these tasks are not a test and that there is right or wrong answer, but that we they are simply asked about their personal perceptions of similarities among molecule representations.

2.3 Dataset

Undergraduate students enrolled in an introductory chemistry course at a large U.S. university were invited to participate in a survey on learning with visual representations. The course had an enrollment of 781 students. Participation was voluntary. Altogether, we collected 26,180 responses from 563 (possibly non-unique) students. 61.6% of the students completed all 50 similarity judgment tasks, on average, students completed 46.5 tasks. Each similarity judgment in response to a triplet comparison task was associated with the feature vectors (x) and molecule vectors (r) of the three molecule representations, as described in 2.1.

3. ANALYSIS

In the following, we describe how we use similarity learning to investigate which visual features drive students’ similarity judgments. We first provide a brief introduction into the metric learning method in general. Then, we describe how we applied this method to our dataset in particular.

3.1 Introduction to Similarity Learning

In general, the goal of similarity learning is to learn a similarity function f that agrees with students’ similarity judgments in the following sense: if item i is judged to be more similar to j than to k, then f(i,j) < f(i,k). The function f can be thought as quantifying the perceived distance or dissimilarity between pairs. Alternatively, the function could quantify the perceptual similarity (inverse distance) between pairs, in which case f(i,j) > f(i,k).

People are better at providing ordinal (i.e., comparative) responses than at providing fine-grained quantitative judgments or ratings [27]. For example, when asked to compare the visual representations in Figure 3, people find it easier to judge whether the target molecule is more similar to the left or the right choice molecule than to judge their similarity on a rating scale. However, it is challenging to machine-learn embeddings from comparisons due to the sheer number of possible triplet comparisons that could be made; the number of distinct triplets is proportional to n³. For example, in our case of n=50 molecule representations, there exist nearly 125,000 distinct triplets. Researchers have observed that while triplet comparisons are easy to answer, they can become tedious and boring after extended sessions [28]. Since we hypothesize that perceived dissimilarities can be accurately represented in d-dimensional space, it is reasonable to conjecture that if the embedding dimension is low (i.e., d << n), then there will be a high degree of redundancy among the triplet comparisons. In fact, researchers have observed that a small subset of these triplet comparisons often suffice to learn a reasonably accurate embedding, lending support to this conjecture [29-31].

3.2 Similarity Learning Approaches

We applied two similarity learning approaches in this paper: similarity learning by ranking [32] and non-metric multi-dimensional scaling. In both cases, we modeled the perceptual similarity between molecules i and j as

\[ S_{ij} = x_i^T A x_j \]

Here A is a symmetric matrix that parameterizes the model. The k, lth element of the matrix, denoted by \( A_{kl} \), represents the importance of the interaction of feature k and feature l in the model. Since we assume A is symmetric, \( A_{kl} = A_{lk} \) and \( S_{ij} = S_{ji} \). Before introducing these approaches, let us define some notation. There are N triplet comparisons. For the nth triplet, let \( i_n \) denote the target-molecule and let \( j_n \) and \( k_n \) denote the two choice-molecules. Let \( y_n \) denote the student’s judgment, specifically \( y_n = +1 \) if the student judged \( j_n \) more similar to \( i_n \) than \( k_n \), and \( y_n = -1 \) otherwise. Each of \( p = 50 \) diagrams also has \( m \) associated features (e.g., numbers of different atoms, bonds, etc.). Arrange the features for each molecule representation into an \( m \times 1 \) molecular feature vector, and the \( m \times 1 \) feature vectors into a \( m \times p \) matrix, \( X \). The nth row of \( X \), denoted \( x_n \), contains the \( m \) features for molecule \( i_n \). The jth row of \( X \), denoted \( r_j \), is a molecule vector for feature \( j \) containing the value of feature \( j \) for all \( N \) representations.

3.2.1 Approach 1: Similarity Learning by Ranking

This approach learns matrix A in our model of perceptual similarity directly from triplet responses via linear regression.

\[ S_{ij} = x_i^T A x_j \]

where \( x_i \) and \( x_j \) are \( m \times 1 \) dimensional feature vectors of the \( m \) features of molecule representations \( i \) and \( j \). The matrix A is \( m \times m \), and the metric learning problem is to estimate \( A \) that minimizes the number of disagreements between the ranking predictions for each triplet (i.e., either \( S_{ij} > S_{ik} \) or vice-versa) and the comparative judgments collected from the students, as proposed by [32].

The first step in this analysis was to estimate \( A \). Formally, the estimation of \( A \) can be written as the following optimization problem. Let \( S \) be the set of all \( n \times m \) symmetric matrices. Solve for \( A \) that minimizes

\[ \hat{A} = \arg \min_{A \in S} \sum_{n=1}^{N} \| y_n - x_i^T A x_j + x_i^T A x_k \|^2 \]

where the superscript T denotes the vector transposition. The matrix \( \hat{A} \) that minimizes the sum of squared errors weights the similarities between the diagram features so as to predict perceptual similarity judgments. In general, the solution \( \hat{A} \) will place some weight on all \( m \) features. We anticipate that the visual features that are not salient do not strongly affect students’ similarity judgments and therefore have lower weights in \( \hat{A} \).

Taking this thinking a step further, we could consider many different optimizations of the type above, where in each case we use different subsets of the features, in order to determine which are most predictive of student judgments. Indeed, some features may be totally irrelevant and worsen, rather than help, the prediction of students’ similarity judgments. Unfortunately, searching over all possible subsets of features is computationally infeasible, so we instead consider the following optimization that approximates this search problem called sparse COMET [33].
\[ \hat{A} = \arg \min_{A \in \mathbb{S}_m} \sum_{n=1}^{N} (y_n - x_n^T A x_{jn} + x_n^T A x_{kn})^2 + \lambda \sum_{k=1}^{m} ||A(k,:)||_2^2 \]

This optimization method uses a cost function that consists of two terms. The first term represents least squares data-fitting cost in the previous optimization. The second term is a Group LASSO penalty, which encourages solutions that have many columns equal to 0. If a column in \( A \) is all zero, then the corresponding feature is not used for prediction. The number of zero-valued columns in the solution depends on \( \lambda > 0 \). Note that we recover the previous optimization when \( \lambda = 0 \). Larger values of \( \lambda \) produce sparser solutions that effectively use fewer features. Features crucial for prediction are excluded only if \( \lambda \) is exceedingly large.

The second step in this analysis was to tune the parameter \( \lambda \) and then to assess the prediction accuracy of our method. To this end, we used 10-fold cross validation. Specifically, we randomly split the complete dataset into 10 equal sized subsets. We removed 2 random subsets as hold-out data and kept the remaining data as training data. We then solved the optimization above with the training data over a range of different \( \lambda \) values. For each \( \lambda \), we scored prediction accuracy on one set of hold-out data to select the optimal value. Then, using our chosen \( \lambda \) value, we solved the optimization again to obtain a final \( A \) using 9/10 of the data, and assessed the prediction accuracy on remaining 1/10 of the data.

The final step was to rank the features based on the weights in matrix. Due to the Group LASSO penalty in the loss function, many of the columns in the resulting matrix are zero. To get the aggregate weight of each relevant feature, we computed the length (norm) of each non-zero column and ranked accordingly.

### 3.2.2 Approach 2: Ordinal Embedding

In this approach, rather than directly making predictions of similarity based on feature vectors and triplet responses, we first used students’ similarity judgments to learn an embedding that spatially represents the similarity of molecule representations as distances in 2-dimensional space. We then identified molecule vectors that account for the distribution of molecular representations in the embedding space.

The first step in this analysis was to learn an embedding. We applied non-metric multidimensional scaling (NMDS) to the 26,180 triplet comparison responses collected from the experiment to learn an embedding of the 50 molecule representations in a two-dimensional space [22]. Embedding in two dimensions allows visualizing the perceived similarity computed by NMDS. The embedding reflects the consensus among students as to which molecular representations were more or less similar. We created 50 different embeddings, using multiple random initializations per embedding in order to account for the non-convexity of NMDS.

The second step was to validate the embedding. To this end, we computed a distance matrix for each embedding. To validate the distance matrices, we used the following cross-validation procedure. We selected 6000 triplet comparison responses uniformly at random to serve as a hold-out dataset. From the remaining triplets, we randomly selected training sets of different size, ranging from 1000 to 20,000 triplet comparison responses. We computed embeddings for each training set. We then used these embeddings and the associated distance matrices to predict students’ similarity judgments. Next, we used the distances in the embedding as a predictor of judgments in the hold-out set; the prediction errors quantify how well the embedding reflects the judgments. We repeated this procedure for training sets of different size. We performed 50-fold cross validation to calculate average prediction error on the learned embeddings. This procedure allowed assessing how prediction performance relates to the training set size (i.e., how many triplets were used to compute an embedding).

The third step in our analysis, after validating our embedding procedure, was to compute an embedding and corresponding distance matrix from the full set of triplets. Since the distance between points in the embedding corresponds to their perceived dissimilarity, we computed a similarity matrix defined as the element-wise inverse of the distance matrix, scaled from 0 to 1.

The fourth step was to identify which features, represented by the feature vectors, drive students’ similarity judgments. Because the embedding was performed in 2 dimensions, we can consider the problem of only choosing 2 feature vectors to combine and compare combinations of pairs of feature vectors to the similarity matrix. For each possible pair, we performed a least squares optimization to find the ideal uniform scaling to match an outer product of our feature vectors to the similarity matrix.

\[ \hat{A} = \arg \min_{A} \sum_{i,j=1}^{P} \left( S_{ij} - x_i^T A x_j \right)^2 \]

subject to \( A_{et} = 0 \) for all \( s,t \) not equal to \( k,l \) or \( l,k \). In other words, only let the \( k,l \) elements of \( A \) be non-zero and optimize these. This equates to fitting \( S \) to the molecule vectors for features \( k \) and \( l \). Here, \( S_{ij} \) represents the value of the perceptual similarity between molecules \( i \) and \( j \) from the embedding. The magnitude of resulting value of \( A_{et} \) tells us how important the interaction of features \( k \) and \( l \) is in representing the similarity. This is basically a correlation coefficient, and it only gauges the marginal value of this interaction (i.e., in isolation of all other interactions). In each case, after learning a matrix \( A \), we computed the corresponding residual value between similarity matrix \( S \) and our combination of 2 features. After performing all possible combinations of pairs of features, we ranked pairs of features in ascending order of residual values, with the smallest residuals being the best approximation of our observed similarity matrix. To evaluate the feature rankings, we used 10-fold cross-validation by performing identical tests on 10 different similarity matrices computed from different embeddings based on equal numbers of triplets to ensure that the original embedding and the non-convexity of NMDS was not a factor in the final ranking of feature pairs.

### 4. RESULTS

#### 4.1 Identifying Important Visual Features

To address research question 1, we used the two similarity learning approaches just described to identify which visual features account for students’ similarity judgments.

##### 4.1.1 Approach 1: Similarity Learning by Ranking

Recall that the first approach entailed learning a similarity function that describes students’ perceived similarity between molecule representations. This approach yielded an average 69% prediction accuracy of students’ similarity judgments (assessed via 10-fold cross validation). This finding indicates that there was consensus over which representations were more or less similar, but also that there were some disagreements among students’ similarity judgments.

To identify which visual features account for students’ similarity judgments, we estimated the weights for each feature in the ma-
Table 1. Top 10 features from the ranking of features with strong weights obtained by Approach 1.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Avg weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distinct letters</td>
<td>4.50%</td>
</tr>
<tr>
<td>Single bonds between Oxygen and Hydrogen</td>
<td>3.45%</td>
</tr>
<tr>
<td>180-degree angle in Hydrogen-Carbon-Fluorine</td>
<td>3.16%</td>
</tr>
<tr>
<td>Double bonds between Oxygen and Nitrogen</td>
<td>3.03%</td>
</tr>
<tr>
<td>Number of Nitrogen atoms</td>
<td>2.99%</td>
</tr>
<tr>
<td>Double bonds between Carbon and Oxygen</td>
<td>2.78%</td>
</tr>
<tr>
<td>120-degree angle in Hydrogen-Carbon-Hydrogen</td>
<td>2.73%</td>
</tr>
<tr>
<td>Number of Oxygen atoms</td>
<td>2.64%</td>
</tr>
<tr>
<td>180-degree angle in Carbon-Carbon-Oxygen</td>
<td>2.62%</td>
</tr>
<tr>
<td>Single bonds between Carbon and Oxygen</td>
<td>2.37%</td>
</tr>
</tbody>
</table>

Table 1 shows the 10 most important features, as determined by a ranking of features according to their aggregate weight computed from matrix A. These results show that the most highly ranked feature is the number of distinct letters, which corresponds to an aggregate educated guess feature. Specific visual features that are relevant to organic molecules were also ranked highly (e.g., the number of single bonds between Oxygen and Hydrogen atoms, the number of bonds between Carbon and Oxygen, the number of Nitrogen and Oxygen atoms). These specific visual features were present in many of the molecules in our dataset. Several visual features also included geometric aspects, specifically bond angles. These features indicate the presence of chemical functional groups that are relevant to predicting molecule’s reactive behaviors.

4.1.2 Approach 2: Ordinal Embedding

Recall that approach s learns an embedding that represents the similarity of molecule representations as distances in a d-dimensional space, from which we then extracted the most important features. First, we established how many dimensions we need to consider (i.e., which d to choose in representing similarity of molecule representations in a d-dimensional space). Using the process of 50-fold cross validation described above, we calculated unit through 20 dimensional embeddings of perceptual similarity. We used 20,000 triplets in this computation to ensure that the number of triplets did not affect the prediction accuracy as the dimension became large. Figure 4 shows that there is no drop in prediction accuracy when embedding in low dimensions versus high, suggesting that perceptual similarity can be accurately represented in a low dimensional subspace, and that there is a high degree of redundancy in the data. This result shows that students’ responses agreed on the relative similarity about 70% of the time.

Next, we generated a 2-dimensional embedding that describes students’ perceived similarity between the molecule representations. Figure 5 shows this embedding, illustrating that molecules naturally form clusters based on their perceptual similarity. These clusters correspond to specific chemical properties shared among the molecules, such as the presence of a particular type of bond or a functional group. We color-coded and labeled some of these clusters to illustrate these characteristics of students’ perceptions. This illustration lends face validity to our embedding approach.

From this embedding, we extracted an ordered list of the feature pairs that best capture students’ similarity judgments, shown in Table 2. The feature pairs in this table were ranked based on how well they approximate the similarity matrix computed from the embedding in Figure 5. The same feature may appear twice in a pair to account for the possibility that a weighted combination of a feature with itself better reflects the observed similarity structure than does a pair of features. In sum, these results show that the most highly ranked features are general visual features, which correspond to the aggregate educated guess features (e.g., number of letters, number of lines). Specific visual features that are relevant to hydrocarbon molecules were also ranked highly (e.g., the number of Carbon and Hydrogen atoms). These specific features were present in many of the molecules in our dataset.

4.1.3 Comparing the Similarity Learning Approaches

While both methods agreed upon the top ranked feature, the similarity learning by ranking approach ranked structural features of the representations that were relevant to hydrocarbons and organic molecules more highly. As the ranking from this method follow predictive power, this ranking indicates that students’ judgments of similarity can best be predicted, and therefore explained, through a combination of the number of different letters and the structural features involving Carbon, Hydrogen, and Oxygen.

4.2 Comparison with “Educated Guesses”

To address research question 2 (do the visual features we identified as salient via metric learning correspond to visual features that students are expected to attend to?), we compared the results from the similarity learning approaches to the educated guess features that we had determined based on the expert-novice litera-
4.3 Number of Similarity Judgments Needed

We addressed our methodological research question 3 (how many similarity judgments we need to assess students’ perceptual knowledge) with the ordinal embedding approach. Specifically, we tested how many triplet comparisons are required to compute a representative embedding of the underlying similarity. Figure 6 shows that gains in prediction accuracy of the embedding were no longer statistically significant beyond 7000 triplet comparisons.

4.4 Differences Between the Two Approaches

The two methods are different and potentially complementary. There is no definitively correct way to fit the common model $S_{ij} = x_i^T A x_j$ to data. The main differences in the final rankings they produce stems from how we are learning matrix $A$ and the restrictions we put on its structure. In approach 1 we are directly working with triplet responses which are perhaps noisy due to disagreements in students’ individual judgments of perceptual similarity, but we are placing fewer restrictions on the learned matrix, allowing for more feature interaction. In approach 2, NMDS is useful for capturing perceived similarity in aggregate, but we enforce much stronger restrictions on the structure of $A$, namely that only two features may interact at once, giving a clearer picture of the importance of a pair of features.

If we had to recommend one approach, we prefer the regression approach (approach 1) because it optimizes prediction error, which is an objective measure of model quality. The embedding approach (approach 2) has its own potential virtues: The low-dimensional embedding provides an implicit form of regularization that may be helpful especially if the amount of response data is small. Also, the embedding provides a visual representation of perceptual similarities which is helpful for model interpretation.

5. DISCUSSION

We applied similarity learning approaches to assess which visual features students focus on when presented with visual representations. We compared two approaches, one that allows us to assess the predictive power of the identified features, and one that allows representing the perceived similarity in a d-dimensional space. Both approaches yield similar results as to which visual features...
are salient to students. Hence, both approaches address research question 1: Which visual features do students focus on when presented with visual representations? We found that students’ similarity judgments of Lewis structures appear to be driven by general visual features such as the number of total and distinct letters, as well as by visual features specific to the types of molecules in our dataset (e.g., number of Hydrogen / Carbon atoms).

Our results also address research question 2: Do the visual features we identified as salient via similarity learning correspond to visual features that students are expected to attend to based on the expert-novice literature on perceptual learning? We found that the identified general visual features align with educated guesses based on the literatures on expertise and perceptual learning, which validates the common “educated guess” approach that instructional designers have to rely on in the absence of assessments of perceptual knowledge. Our results also suggest that, in addition to these general features, students learn to pay attention to key visual features that are highly domain-specific; such as features that indicate the presence of functional groups that are predictive of chemical behaviors. Furthermore, our results show that a few key features predict students’ perceptions of similarity between visual representations with accuracy of about 70%.

Finally, we addressed our methodical research question 3: How many similarity judgments do we need to assess students’ perceptual knowledge? Our results show that about 7,000 responses to triplet comparison tasks are sufficient in assessing a population’s perceptual knowledge. Using a survey with 50 triplet comparison tasks (as in our experiment), that means an N of 140 participants will yield valid assessments of perceptual knowledge.

6. LIMITATIONS

Although both similarity learning approaches had rigorous theoretical backing, we made a few assumptions about our triplet comparison data that had inherent limitations of note. In both of these methods, we are not modelling individual students, but rather the population as a whole. Consequently, we assume that the triplets and therefore the judgments of similarity are independent of one another. This assumption allows us to learn the rankings of features and feature pairs for the students’ collectively, but it does not provide a ranking for an individual. Further, because judging similarity representations is a subjective task, students’ judgments may in certain cases conflict with one another. Even with an extremely large number of similarity judgments, complete consensus is unlikely, and therefore, perfect prediction of student judgments is similarly difficult to achieve. Hence, future research needs to investigate how to expand the present approach to modeling individual perceptual knowledge.

Another limitation pertains to the ordinal embedding procedure. For visualization purposes, we embedded the molecules into a 2-dimensional space. Higher dimensional embedding may more accurately capture perceptual dissimilarities. Future research should explore this question.

7. FUTURE DIRECTIONS

We will expand our research to other types of visual representations typically used in chemistry instruction (see Figure 1). Further, we will gather data from expert chemists and compare them to data from novices and advanced learners. Based on this comparison, we will identify a “perceptual knowledge gap” between students and experts. Specifically, we will identify visual features that experts attend to but students do not.

Further, we will expand similarity learning so that it can assess an individual student’s perceptual knowledge in real time. The current approach is limited in that it requires a large number of similarity judgments to assess students’ perceptual knowledge, which is only feasible if we are interested in assessing perceptual knowledge of a population of interest (e.g., novices, advanced students, experts), and because we assume independence among similarity judgments. To address this limitation, we will combine our similarity learning approach with cognitive modeling methods (e.g., Bayesian knowledge tracing). For example, a similarity judgment survey may provide a prior for in a cognitive model, and students’ performance on perceptual learning tasks may inform the choice of representations for a small number similarity judgment tasks interspersed in the learning activity.

This expansion will provide the basis for the design of adaptive instruction for perceptual knowledge that can provide appropriate sequences of perceptual learning tasks that draw students’ attention to visual features they yet have to learn. Further, knowing which visual features students have not yet learned can serve as a basis for the design of visual feedback that highlights visual features when students make mistakes on perceptual learning tasks.

In sum, we will use the similarity learning approach described in this paper both to design instruction for perceptual learning and to assess perceptual knowledge as a learning outcome.

8. CONCLUSIONS

This paper described a new approach to assess students’ perceptual knowledge. We used this approach to validate the “educated guesses” approach. In addition, we offer more formal pathways for instructional designers to create perceptual learning assessments. Because developing adaptive instruction for perceptual knowledge relies on such assessments, this paper makes an important contribution to cognitive modeling research.

This paper also makes important contributions to machine learning. We provide a new mathematical approach to quantify the accuracy of perceptual embeddings learned from similarity judgments. Specifically, we derived bounds on the accuracy of embeddings learned from small numbers of comparative judgments by adapting recently developed large-sample analysis methods [34]. This approach provided new algorithms for generating embeddings that are provably accurate. We investigated new methods for embedding based on spectral methods inspired by spectral ranking algorithms [35]. Our experiment yielded an empirical validation with perceptual data from undergraduates, as well as new machine learning methods to assess how visual features predict or encode perceptual similarity judgments. Specifically, we explored the application of group Lasso algorithms for automatically selecting the most perceptually salient features [36]. Our experiment empirically evaluated the group Lasso approach.

In sum, our work provides a crucial stepping stone towards adaptive instruction for perceptual knowledge. Perceptual knowledge is by definition implicit and does not lend itself to the kinds of techniques used in traditional cognitive modeling approaches (e.g., think-alouds, interviews). We presented and evaluated two similarity learning approaches that can determine which visual features students attend to when perceiving visual representations.

9. ACKNOWLEDGMENTS

We thank Professor John Moore for his help in recruiting participants for this study, and the LUCID group for their suggestions.

10. REFERENCES

in Gilbert, J. et al. (Eds.): 'Visualization' (Springer, 2008), pp. 191-208


ABSTRACT
The Cognitive Tutor Algebra I (CTAI) curriculum, which includes both textbook and online components, has been shown to boost student learning by about 0.2 standard deviations in a randomized effectiveness trial. Students who were assigned to the experimental condition varied substantially in how, and how much, the used the online component of CTAI, but original analyses of the experimental data focused on estimating average effects, and did not examine whether the CTAI treatment effect varied by the amount of style of usage. This study leverages log data from the experiment to present a more nuanced analysis. It uses the framework of Principal Stratification, which estimates the varying CTAI treatment effect as a function of “potential” usage—either how students used the program, or how they would have used it had they been assigned to the treatment condition. With experimental data, Principal Stratification does not require that we assume that all relevant variables have been measured. With this framework, we find that students who receive a medium amount of assistance from the software (in the form of hints and error feedback) experience the largest effects, with lower effects for students who receive a lot or a little; and evidence that students who do not follow the curriculum order experience smaller treatment effects.

Keywords
Causal Mechanisms, Principal Stratification, Intelligent Tutors, Bayesian Hierarchical Models

1. INTRODUCTION
Intelligent tutors—computer programs designed to teach—claim to improve student achievement via a number of mechanisms, including a reliance on cognitive modeling, instant feedback, and individualized instruction. As the demand for intelligent tutors grows, so does the demand for evidence of their effectiveness, and the educational research community has kept apace, with a number of randomized field trials [e.g. 5, 9, 14]. Since intelligent tutors are computerized, it is relatively easy for experimenters to collect student log data, alongside traditional evaluation data. This paper will provide a template for how to evaluate the log data from an intelligent tutor experiment, to help elucidate the intelligent tutors’ mechanisms and when and for whom they work.

A recent randomized study of Carnegie Learning’s Cognitive Tutor Algebra I (CTAI) curriculum, under real-life conditions, was reported in [8]. In the second year of the experiment, in high school classrooms, the study found, that CTAI boosts student learning by about 0.2 standard deviation, on average. However, in the first year of the experiment CTAI’s effect was close to nil. Surely one explanation for this heterogeneity is that students and teachers used the curriculum differently in the two years—but how? What aspects of student usage predict a treatment effect?

The effectiveness trial produced extensive student usage data, as the computer program logged students’ activity. In this paper, we use this data—in particular, usage data from the 2nd-year high school sample that apparently experienced a substantial CTAI effect—to explore the relationship between student usage and causal effects. In future work, we will attempt to use these findings to explain the difference between the two years of the experiment.

A preliminary study, [17], argued that the best causal model for the usage data relies on the “principal stratification” framework [2, 7], under which students who used the CTAI software in a particular way are compared to control students who would have used it in the same way, had they been assigned to treatment. This study is the first full study that last year’s preliminary study promised. It provides two sets of results exploring different aspects of CTAI’s mechanisms: an analysis of assistance, which is calculated from the hints that students request and the errors they make, and an analysis of the the order at which students work on CTAI’s sections. The paper also includes a more detailed discussion of the models, and a discussion of some issues with the results in [17].

2. DEFINING THE QUESTION: HOW DOES POTENTIAL USAGE MODERATE THE CTAI EFFECT?
As in [17], in this paper we model student usage under the principal stratification (PS) framework, a generalization of the Neyman-Rubin Causal Model [15] of potential outcomes. If Z is a binary treatment assignment, and Y
is an outcome, each subject has two potential outcomes: Y(Z = 1) and Y(Z = 0), the outcome she would present under the treatment condition, and under the control condition, respectively. Each of these is defined, though unobserved, prior to treatment assignment Z. After subjects have been assigned to treatment, exactly one of the potential outcomes is observable for each subject: for treatment subjects, the observed Y = Y(Z = 1), and for control subjects, Y = Y(Z = 0).

[2] generalized the potential outcomes framework, introducing the concept of principal strata. A principal stratum is a grouping of subjects based on potential values of intermediate outcomes. For example, if we call students’ usage values U, each student has usage values U(Z = 1) and U(Z = 0)—the usage they would exhibit under the treatment and control conditions, respectively. In the CTAI experiment, U(Z = 0) = 0 for all subjects, since no control subjects had access to the cognitive tutor. Say we model usage as a categorical value for K categories, U = 1,...,K. Then there are k principal strata: {U(Z = 1) = k, U(Z = 0) = 0} for k = 1,...,K. In this framework, principal stratum membership is observed for subjects in the treatment group—we observe their usage once they are assigned to treatment, and we know from the experimental design that they would not have used the tutor had they been assigned to control. The potential usage for students in the control group, however, is unobserved, and must be estimated; the following section will discuss this process in more detail.

For each stratum, we can define a “principal effect”: the average treatment effect \( \tau_k = \mathbb{E}[Y(Z = 1) - Y(Z = 0) | U(Z = 1) = k, U(Z = 0) = 0] \) for subjects in principal stratum k. Although unobserved, these strata are defined prior to treatment assignment—if assigned to treatment, what would a student’s usage be? That is, observed usage U is an intermediate outcome, or a mediator, but potential usage U(Z = 0) and U(Z = 1) is a pre-treatment covariate, or a moderator. The principal effects are, then, subgroup effects, for various levels of potential usage. Differences between principal effects are differences in the effect of CTAI for students who use (or would use) CTAI differently. To put it more precisely, consider the difference \( \tau_1 - \tau_k \). This is the difference in the effect of CTAI between the group of subjects who, if given the opportunity, would exhibit usage in the amount of j or the amount of k. While the effect estimates \( \tau_1 \) and \( \tau_k \) are themselves causal (due to randomization) the difference between them could be due to the effect of usage, or to pre-treatment differences between students in the two groups. In other words, since usage values were not assigned randomly, the difference in CTAI effect between two usage principal strata are not necessarily causal. Still, estimating principal effects, and their differences, along with differences in the composition of principal strata, can shed light on the mechanisms of CTAI.

In one of our analyses below, usage is measured as a continuous, not categorical, variable, so the PS approach entails discretizing usage scores. [4] suggested an alternative: modeling potential usage as a continuous mediator, via an interaction in a regression analysis. They refer to this analysis as a “causal effect predictiveness” or CEP curve. CEP curves are directly analogous to principal strata effects, but with continuous intermediate variables.

3. ESTIMATING PRINCIPAL EFFECTS AND CEP CURVES

Estimating principal effects and CEP curves is a complex process, since first we must estimate unobserved principal strata membership or potential usage variables, and only then to estimate treatment effects. In fact, principal effects, in some circumstances, are only partially identified—even in an infinite sample, a Bayesian credible interval for a principal effect may have a finite width. This is especially the case when researchers attempt to estimate principal effects without covariates, and while relaxing traditional instrumental variables assumptions. However, in the presence of covariates that predict usage variables, we may estimate informative effects.

This section describes the models that we use to estimate principal effects and CEP curves. More details can be found in [16].

3.1 The Model

In general, the central challenge in PS modeling is that principal strata membership is unknown. In the CTAI experiment, since control students had no access to CTAI software, strata membership for the treatment group is known, but must be estimated for the control group. The distribution of the potential outcomes for Y, conditional on covariates, \( p(Y(Z = 0)|X_i) \), can be decomposed into the probability distribution of Y given \( U_i(Z = 1) \), which is the distribution of interest, times the distribution of \( p(U_i(Z = 1)|X_i) \), which, due to random assignment, may be estimated from the treatment group. Then, we may estimate the parameters of \( p(Y(Z = 0)|U(Z = 1) = a, X) \) and compare them to the analogous distribution \( p(Y(Z = 1)|U(Z = 1) = a, X) \) yielding estimates of treatment effects within principal strata.

If we assume that outcomes are conditionally normally distributed, the result is a finite normal mixture model:

\[
p(Y_i(Z = 0)|X_i) = \sum_{k=1}^{K} \Pr(U_i(Z = 1) = k|X_i) \phi(\mu_k(Z = 0) + f_k(X_i), \sigma_k) \tag{1}
\]

and

\[
p(Y_i(Z = 1)|X_i, U(Z = 1) = k) = \phi(\mu_k(Z = 1) + f_k(X_i), \sigma_k) \tag{2}
\]

where \( \phi(\mu, \sigma) \) is the normal density with mean \( \mu \) and standard deviation \( \sigma \). Equations (1)-(2) additionally assume no interaction between covariates and treatment status within principal strata. The contribution of covariates \( X_i \) to the mean of \( Y_i(Z = 1) \) can vary from stratum to stratum, but within stratum it does not vary with treatment status. In practice, we estimate \( f_k(X_i) \) as linear in covariates:

\[
f_k(X_i) = X_i^T \beta_k \tag{3}
\]

where we estimate a different set of slopes \( \beta \) in each stratum \( k \). The linearity assumption can be relaxed or adjusted based on the model’s fit to the data. The effect of CTAI in the \( k^{th} \) principal stratum is \( \tau_k = \mu_k(Z = 1) - \mu_k(Z = 0) \).
The model to estimate a CEP curve is broadly similar to the PS model, with one important difference. In the PS model, usage was parametrized as a categorical variable, and different effects were calculated for each stratum. In the CEP framework, usage is continuous, and its interaction with the effect of treatment must be modeled. As the next section will discuss, we chose to model the CTAI effect as quadratic in usage, for instance. The CEP outcome model, then, is

\[
p(Y_i(Z = 0)|X_i) = p(U|Z = 0)(a)\phi(f_U(Z = 0) + f_X(X_i), \sigma). \tag{4}
\]

and

\[
p(Y_i(Z = 1)|X_i, U(Z = 1) = a) = \phi(f_U(Z = 1) + f_X(X_i), \sigma). \tag{5}
\]

where \(p(U|Z = 1)|X(a)\) is the density of \(U(Z = 1)\) conditional on \(X\). \(f_U(Z = 0)\) and \(f_U(Z = 1)\) are parametric functions of usage for treated and untreated subjects, respectively, and \(f_X(X_i)\) is a model for covariates. The CTAI treatment effect is now a function of potential usage, \(U(Z = 1): \tau(a) = f_U(Z = 1) - f_U(Z = 0)\).

Models (1), (2), (4), and (5) all require a model for the density of usage, as a function of covariates \(X\). In our paper, the usage model, \(p(U(Z = 1)|X)\), is also linear in \(X\). When the the usage variable is continuous, it is:

\[
p(U(Z = 1)|X) = \phi(X_\gamma, \sigma_U) \tag{6}
\]

normal-theory linear regression. In PS models, when we discretize \(U\), we do so after fitting model 6.

When \(U\) is binary, we use a linear logistic regression to estimate \(p(U(Z = 1)|X)\):

\[
Pr(U(Z = 1)|X) = invLogit(X_\gamma) \tag{7}
\]

We fit all of the above models simultaneously with Markov Chain Monte Carlo (MCMC), using JAGS and R [10, 11]. Since MCMC is a Bayesian technique, it required priors; we put a normal prior with mean zero and standard deviation 3 on each of the model fixed effects—a prior that easily accommodates any plausible effect, but discourages outlandish estimates. We put a weakly-informative inverse-gamma(0.001, 0.001) prior on the variance parameters.

The models for assistance, described below in Section 5, were fit with the Stampede Supercomputer at the Texas Advanced Computing Center.

### 3.2 Some Potential Pitfalls

[17] presented a set of preliminary results from principal stratification analyses. They were presented as a first attempt at fitting principal stratification models, to illustrate the technique and its potential for helping us understand some of the factors behind CTAI’s effect. However, since the EDM 2015 conference, a number of issues emerged with the preliminary results in that paper. It is instructive to discuss those results as an illustration of potential pitfalls in principal stratification analysis.

#### 3.2.1 Model Convergence

One of the first checks of a Markov Chain Monte Carlo model is convergence. MCMC models (ideally) proceed through two stages: first, in the “burn-in” stage, parameter estimates fluctuate widely as the model converges on the posterior distribution for the parameters. After convergence, the algorithm draws from the posterior distribution of the parameters. From these draws, we can estimate the posterior’s mean—a point estimate for the parameters—standard deviation, and quantiles. However, it is not always clear when the burn-in period has ended, and the model has begun sampling from the posterior. There are two principal ways of checking this. Both methods rely on running the MCMC separately in two or more chains. That is, start the Gibbs sampler \(c\) separate times, with \(c\) sets of starting values for the parameters, and let the \(c\) separate chains each take their own course. Then, the results from the \(c\) chains may be compared; if the model has converged, they should resemble one another, since they each would have converged on the true posterior distribution. One method of measuring whether this is the case is the Gelman-Rubin R-hat statistic, which compares the within-chain variance two the between-chain variance; since, after the burn-in stage, the chains should all be sampling from the same distribution, the between-chain variance should be small. At convergence, the R-hat statistic should be approximately one. Typically, values of R-hat less than 1.1 are acceptable. Additionally, analysts may inspect “traceplots”: plots of the \(c\) chains for each parameter. If the chains are each stationary—that is, not changing in location or variance—and seem to share a location and scale with each other, the model has most likely converged. If the various chains converge on different distributions, the model might be non-identified, or multi-modal—several different estimates might be equally consistent with the data.

Some of the models in [17] may not have achieved convergence. In this paper, all of the models had clearly achieved convergence.

#### 3.2.2 Gain-Score Modeling and Covariate Selection

A second concern with the model results from [17] emerged from our use of gain-scores—the difference between a post-test and a pre-test—as the outcome in the model, as opposed to the post-tests themselves. The problem with doing so is that the usage model was linear in the pre-test, by design. In the assistance model, for instance, assistance is anti-correlated with pretests, so the the control subjects who were estimated to have high levels of potential assistance also had high pre-test scores. On the other hand, pre-test scores are anti-correlated with gain scores, due to regression to the mean. So the control subjects with high estimated assistance will have lower gain scores on average. This can lead to an overestimate of an effect in the high-assistance stratum, especially if the usage model is misspecified. In principle this is an easy problem to correct, simply by including pre-test scores as a covariate in the outcome model as well. However, doing so would undermine the rationale of gain score modeling. For these reasons, we relied exclusively on post-test modeling in this paper, with the pre-test as a covariate in both the usage and outcome sub-models.

#### 3.2.3 Student-Level Averages as Usage Variables

[17], and an earlier version of this manuscript, estimated the variation of the CTAI effect as a function of the av-
verage number of hints and errors each student requested or committed (called “assistance”). These averages were taken over all of each student’s worked problems. Subsequent analysis revealed a curious phenomenon: the students with the most extreme average assistance values worked very few problems—almost uniformly so. Interpreting the CEP curve, in this case, becomes nearly impossible, since average assistance is so closely related to the amount of usage. The reason for the close relationship is straightforward: sample averages are random variables, and the variance of a sample mean is directly proportional to the sample size. The average assistance values for the group of students who worked very few problems had a high variance; conversely, the variance of average assistance for students who worked a large number of problems was much smaller.

The solution we chose for this issue was to run the model not on student-level average assistance values, but on problem-level data directly, adding another level into the multilevel structure. That way, the model considers student-level usage variables to be latent, as opposed to manifest (i.e. directly observed). Extreme values of latent variables estimated from a small number of problems enter into the model less as students with extreme usage patterns, and more as students whose usage is poorly-determined. In other words, from one MCMC draw to another, the estimate for each low-usage student’s assistance value would vary considerably, so low-usage students would contribute little to the overall estimate of the CEP curve. We discuss the problem-level assistance model in Section 5.

3.2.4 Model Validation
The difficulty of constructing correct principle stratification models, and the ease of constructing models that yield misleading results, suggests that PS models should undergo rigorous specification checking before they are believed. [1], an excellent example of careful principal stratification analysis, provided guidance on how to validate a PS model, which we followed. We conducted three types of checks with each model:

- Estimating each effect with multiple different models and checking for concordance. In the assistance analysis, we estimated MCMC models treating the usage variable as either categorical or continuous. In both analyses we estimated both a normal-distribution model, as discussed in in Section 3.1, and a “robust” model, in which we substituted student’s t-distribution for normal distributions in the model, allowing for outliers.
- Inspecting residual plots to assess model fit, for both the usage model and the outcome model.
- Estimating models with made-up outcome data. We did this primarily with a placebo outcome, generated by adding random noise to the pre-test variable. We then hoped not to find any treatment effects.

Unfortunately, we cannot claim, at this point, that a method or model exists that will always recover the correct answer and never mislead—each model needs to be carefully tailored to its data, and then validated.

4. THE DATA
The CTAI experiment is described in [8]. The study was conducted in 73 high schools and 74 middle schools in 52 urban, suburban, and rural school districts in seven states, encompassing nearly 18,700 high school students and 6,800 middle school students. The schools were matched on a set of covariates prior to randomization, and were subsequently randomized to treatment or control conditions within matched pairs.

The study was an effectiveness trial, where the intervention must be adopted in as naturalistic conditions as possible. This means the study is supposed to capture common implementation variation resulting from imperfect implementation or even refusal to implement certain instructional materials. The naturalistic design of the experiment is particularly important for our analysis of student usage—usage patterns in the experiment plausibly correspond with what we may expect in general.

For the current study, we used only data from the second cohort in high schools. This is because that was the stratum in which overall effects were detected at the 5% level. Indeed, in the first year of implementation point estimates for the effect were close to zero. It may be the case that the difference in effect between the first and second years (a difference which itself is statistically significant) is due to different usage patterns. We hope that our larger project of estimating treatment effect heterogeneity by usage will help explicate the heterogeneity by cohort.

Software usage data is available for only a subset of the students in the treatment group. Considering only students who were present at post-test and are thus a part of outcomes analyses, we have usage logs for 83%. Students not present at post-test are considered to have attritted from the study.

The percentage of non-attribitted students for whom we have usage data varies by school, from 0% (n=3 schools) to 100% (n=20 schools). We assume that schools that have 0% coverage did not implement the CTAI curriculum, despite being assigned to the treatment group. Carnegie Learning was unable, for technical reasons, to retrieve software usage log data for that school.

4.1 Imputing Missing Data
As described above, there were missing data values in the covariates, as well as in the student log scores. We used the missForest package in R [18, 11] to impute missing covariate values. The out-of-box normalized root mean-squared-error for the imputation was 0.02. Since this value is so low, since there was a relatively small amount of missing data, and since covariates play a merely predictive role in our analy-
sis, we assumed that the uncertainty from other aspects of the model would dominate the uncertainty due to covariate imputation and only imputed one dataset, rather than a full multiple imputation.

Missing usage data presents a more serious problem. First, some schools in the CTAI study were not included in the usage dataset. We deleted these schools from the analysis, along with their matched pairs. Since a matched randomized experiment is an aggregate of a randomized trial in each matched pair, discarding the matched pairs with missing data is nearly benign.

We classified within-school missing usage data into two groups: some students did not have usage data because they did not use the software. Since absolute software usage is driven primarily by teachers, we calculated the proportion of students with missing data for each teacher. If almost all of a teacher’s students were missing from the usage dataset, we assumed that they did not use the tutor in their classroom.

The rest of the missing student usage data was due to our inability to match students to their records. We assumed that these data were missing at random [6]—that their missingness was ignorable conditional on their measured covariates. The missingness was likely not missing completely at random, since students who were difficult to match generally did not fill out their student information thoroughly, and thoroughness may correlate with post-test scores or usage patterns. The imputation strategy for these missing data points was identical to the imputation of unobserved potential usage for the control students. That is, the same model that estimated densities for usage variables for control students also estimated missing usage data for some treated students. The missing data strategy in this case was, therefore, either full-information maximum likelihood or MCMC, depending on the analysis.

5. HINTS AND ERRORS

5.1 Assistance Scores

<table>
<thead>
<tr>
<th></th>
<th>#Errors=0</th>
<th>#Errors&gt;0</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Hints=0</td>
<td>0.42</td>
<td>0.34</td>
<td>0.76</td>
</tr>
<tr>
<td>#Hints&gt;0</td>
<td>0.01</td>
<td>0.23</td>
<td>0.24</td>
</tr>
<tr>
<td>Sum</td>
<td>0.43</td>
<td>0.57</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 1: The proportions of problems in our dataset in which students make at least one error or request at least one hint.

[12] defined assistance as the sum of the number of hints students request and the number of errors they make, which together represent the feedback CTAI gives the students. High assistance indicates that a student is struggling.

Hints and errors vary from problem to problem, from section to section, and from student to student. Table 1 shows the joint probability of requesting at least one hint and making at least one error in our dataset. In 58% of worked problems, the student requested at least one hint or one error. Further, hints and errors tend to accompany each other: in only 1% of worked problems the student requested a hint without making an error. In many problems, hints and errors occur sequentially: a student will work part of the problem, perhaps make an error and receive feedback, perhaps request a hint, and then move on to the rest of the problem. It is important to keep in mind, then, that hints do not always precede errors—sometimes, they are the result of a prior error made while working the same problem.

Figure 1 plots the average number of hints a student requests as a function of the average number of errors he makes. While most students request between 0 and two hints per problem, and make between one and eight errors per problem, some students request far more hints or make far more errors. Further, students who request more hints are much more likely to make more errors. The size of the points in Figure 1 is proportional to the square root of the number of problems they completed—and hence to the standard deviation of the plotted averages. The extreme values in the figure typically come from students who work very few problems, as described in Section 3.2.3, complicating the interpretation of a model that uses average hints or errors as a mediator variable.

For that reason, we incorporated a problem-level sub-model for assistance into our larger principal stratification model. Rather than model the total number of hints and errors per problem, which would necessitate a complex, and possibly misspecified, count-data model, we modeled the probability of a student requesting a hint or making an error (or both) on each problem. The model was as follows:

$$Pr(A_{ip} \geq 1) = \text{invLogit}(U_i + \delta_{i[p]})$$

Where $A_{ip}$ is the total amount of assistance, i.e. hints and errors, that student $i$ experiences from problem $p$. $U_i$ is a
random student effect, representing the student’s propensity to receive assistance on a problem, and $\delta_s[p]$ is a section random effect.\(^2\)

The variable $U_i$, student $i$’s “assistance score,” is the mediator that we use to predict her CTAI treatment effect.

$U_i$ is itself predicted, in turn, by a set of covariates including pretest scores, demographics, and teacher random effects nested within school random effects. The results of this usage model are available upon request. They show that prior test scores and “gifted” status are inversely correlated with assistance scores—higher performing students are less likely to make errors or request hints. Special education students are more likely to receive assistance, and males are less likely than females.

There are a number of ways to interpret these results. The results reflect varying CTAI effects for various usage patterns. One of CTAI’s selling points is the instant feedback it provides students as they work through and complete problems. Students who under-utilize this service—in the low assistance stratum—are then likely to experience a smaller CTAI effect. This may be because they began as excellent students—assistance is anti-correlated with pretest scores—and hence did not need the extra help that CTAI provides. Alternatively, students with low assistance scores may be under-utilizing the service for a different reason; perhaps they feared that requesting too many hints, or making too many mistakes, would slow their progress through the tutor, so they were overly cautious.

Students who request hints or make errors quickly, without slow deliberation, may not be able to learn from the problems they work. Some students “game” the system, by requesting hints until they are provided with the correct answer, or they simply do not try very hard to figure out the answer themselves. It may be that the students in the CTAI experiment with very high assistance scores, experience lower treatment effects for some of these reasons. Alternatively, they might have struggled with the material in general, and required more personalized help from a teacher, as opposed to a computerized tutor.

However, students in the middle of the assistance distribution experienced large CTAI effects, suggesting an assistance “sweet spot.” In future trials, teachers could be instructed to encourage their students to use a medium number of hints, and complete problems with a moderate amount of caution—trying hard to answer problems correctly, but also allowing themselves to make mistakes. If this strategy leads to higher CTAI effects, it suggests that part of the CTAI effect heterogeneity across usage patterns is causal—that using the system differently leads to higher effects.

### 6. SKIPPING SECTIONS

An important part of the design of CTAI is the scaffolding of skills and knowledge. The skills that students learn in Algebra I build on each other, so the order in which students learn material and master skills matters—at least in theory. The design of CTAI accounts for this order, by insisting that students master certain skills before moving on to others. Indeed, that is the notion that lies behind the sections of the CTAI curriculum.

We attempted to test the hypothesis that this scaffolding matters—that is, do students who the CTAI curriculum learn more from CTAI than students who do not? To answer
this question, we compared the order in which students in the CTAI experiment worked on sections to the intended order. About 80% of students worked on the sections in order. However, 20% of students skipped at least one section. Did the students who skipped one or more sections experience the same CTAI effect as those who completed the sections in the intended order? More precisely, is the CTAI effect the same in the principal stratum of students who, if assigned to CTAI, would complete the section in order, and in the principal stratum of students who, if assigned to CTAI, would skip at least one section?

A complication in estimating counterfactual stratum membership for control students in this case was that in the CTAI setup, teachers, not students, control which sections the students work on. Indeed, there were 38 teachers in the treatment group for whom we had data on whether students skipped a section. Of those 38 teachers, 17 teachers did not have any students who skipped any sections at all, while there were five teachers more than 80% of whose students skipped sections. Since such a large proportion of the variation in section-skipping occurred at the teacher level, we included a set of teacher-level predictors in our usage model. An anonymous reviewer alerted us to the threat of over-fitting; hence, due to the small number of teachers in the treatment group, we chose only two teacher level covariates in the model: percent ESL, and average pre-test. The small covariate-to-sample size ratio at both the student and the teacher levels, combined with the informative priors [See 3], should alleviate concerns of over-fitting.

The usage model, whose results are available upon request, was unsuccessful in estimating precise effects for any covariate, but in aggregate was able to predict stratum membership. One exception is that students with higher pretest scores are more likely to skip sections, as are teachers whose students have higher pretest scores on average.

<table>
<thead>
<tr>
<th>Stratum</th>
<th>Effect (Normal)</th>
<th>Effect (Robust)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do Not Skip</td>
<td>0.27</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>(0.09 – 0.44)</td>
<td>(0.05 – 0.33)</td>
</tr>
<tr>
<td>Skip ≥ 1 If Treated</td>
<td>-0.09</td>
<td>-0.07</td>
</tr>
<tr>
<td></td>
<td>0.13</td>
<td>0.11</td>
</tr>
<tr>
<td>(0.33, 0.17)</td>
<td>(-0.32, 0.48)</td>
<td></td>
</tr>
<tr>
<td>Difference</td>
<td>-0.36</td>
<td>-0.26</td>
</tr>
<tr>
<td></td>
<td>0.12</td>
<td>0.11</td>
</tr>
<tr>
<td></td>
<td>(-0.59, -0.12)</td>
<td>(-0.48, -0.03)</td>
</tr>
</tbody>
</table>

Table 2: The CTAI effect in the two principal strata defined by whether a not a student would skip a section if they were assigned to the treatment. We estimated principal effects with both an MCMC model based on the normal distribution, based on the more robust student’s t-distribution. Standard deviations of the posteriors are in italics, and 95% credible intervals (MCMC) are provided in parentheses under the estimates.

The results of our analysis are in Table 2 and Figure 3. Both models detect significantly greater treatment effects in the principal stratum of students who would not skip sections if assigned to the treatment, than in the stratum of students who would. This might be taken as evidence that the order in which students complete sections plays a large role in the effectiveness of CTAI. Alternatively, it may be that teachers who tinker with the order of sections that their students work are likely to tinker with other aspects of the CTAI design as well, to deleterious effect (perhaps along the lines of [13]). In either reading, the effect of CTAI is not merely due to the practice it gives students, or immediate feedback, but also to its underlying pedagogical and cognitive theory.

A third possibility is that the entire difference is driven by an underlying teacher or student characteristic, such as ability; students with higher pretest scores are more likely to skip sections—perhaps the treatment effect is significantly lower for them, as well.

7. DISCUSSION
We showed that without additional identification assumptions, researchers can use log data to form a deeper understanding of their software’s effect. However, we also discussed some of the difficulties in estimating these models correctly.

We updated and clarified a result from our preliminary study [17]. We find that the relationship between the amount of assistance students receive from CTAI and the CTAI treatment effect they experience is not monotonic. The highest effects appear for the students who receive a medium amount of assistance; those who receive much more or less experience smaller treatment effects, on average. This may be the result of student attributes—that the students at the margins are either too advanced or gaming the software—or it may be that certain modes of software usage are better than
others.

Next, we investigated if students who skip a section in the recommended curriculum, working on sections out of order, may experience lower effects. The result may confirm part of the motivating theory behind CTAI: that Algebra I skills build on each other, so the order at which students work on material can contribute or detract from their success.

Along those lines, we plan a number of future analyses. We hope to update the preliminary study’s results that suggested that the CTAI treatment effect increases with the amount of usage, and to investigate the dependence of the CTAI effect on students’ mastery of sections. Further along, we hope to discover and define interesting multivariate principal strata, perhaps as the result of a cluster analysis of the high-dimensional usage data.

Finally, after cultivating a more complete understanding of the usage patterns that lead to higher CTAI effects, we can explore treatment-effect heterogeneity. In particular, we may be able to answer why in the first year of implementation CTAI did not seem to boost test scores, but in the second year it did. Was differential usage to blame?

In the meantime, this paper uses rigorous causal methods to confirm some previous hypotheses about CTAI’s causal mechanisms, and points a way forward for future work modeling usage variables in experimental designs.

8. ACKNOWLEDGMENTS

This work is supported by the United States National Science Foundation Grant #DRL-1420374 to the RAND Corporation and by the Institute of Education Sciences, U.S. Department of Education, through Grant R305B1000012 to Carnegie Mellon University. The opinions expressed are those of the authors and are not intended to represent views of the Institute or the U.S. Department of Education or the National Science Foundation. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper. http://www.tacc.utexas.edu. Thanks to Steve Fancsali, Steve Ritter, and Susan Berman for processing and delivering the CTAI usage data. Thanks to Brian Junker for helpful advice and guidance.

References


LIVELINET: A Multimodal Deep Recurrent Neural Network to Predict Liveliness in Educational Videos

Arjun Sharma
Xerox Research Centre India
Arjun.Sharma@xerox.com

Arijit Biswas
Xerox Research Centre India
Arijit.Biswas@xerox.com

Ankit Gandhi
Xerox Research Centre India
Ankit.Gandhi@xerox.com

Sonal Patil
Xerox Research Centre India
Sonal.Patil@xerox.com

Om Deshmukh
Xerox Research Centre India
Om.Deshmukh@xerox.com

ABSTRACT
Online educational videos have emerged as one of the most popular modes of learning in the recent years. Studies have shown that liveliness is highly correlated to engagement in educational videos. While previous work has focused on feature engineering to estimate liveliness and that too using only the acoustic information, in this paper we propose a technique called LIVELINET that combines audio and visual information to predict liveliness. First, a convolutional neural network is used to predict the visual setup, which in turn identifies the modalities (visual and/or audio) to be used for liveliness prediction. Second, we propose a novel method that uses multimodal deep recurrent neural networks to automatically estimate if an educational video is lively or not. On the StyleX dataset of 430 one-minute long educational video snippets, our approach shows an relative improvement of 6% and 1.9% compared to a multimodal baseline and a deep network baseline using only the audio information respectively.

Keywords
Liveliness, Educational Videos, Recurrent Neural Network, Deep Learning, LSTM, Engagement, Multimodal Analysis.

1. INTRODUCTION
The amount of freely available online educational videos has grown significantly over the last decade. Several recent studies [1, 2, 3] have demonstrated that when educational videos are not engaging, students tend to lose interest in the course content. This has led to recent research activity in speech analysis of educational videos. Authors in [4] used crowd-sourced descriptors of 100 video clips to identify various speaking-style dimensions such as liveliness, speaking rate, clarity, formality etc. that drive student engagement and demonstrated that liveliness plays the most significant role in video engagement. Using a set of acoustic features and LASSO regression, the authors also developed automatic methods to predict liveliness and speaking rate. The Authors in [5] analyze the prosodic variables in a corpus of eighteen oral presentations made by students of Technical English, all of whom were native speakers of Swedish. They found out that high pitch variation in speech is highly correlated with liveliness. Arskiere et al. [6] built a large scale educational video corpus called StyleX for engagement analysis and provided initial insights into the effect of various speaking-style dimensions on learner engagement. They also found out that liveliness is the most influential dimension in making a video engaging. In this paper, we propose a novel multimodal approach called LIVELINET that uses deep convolutional neural networks and deep recurrent neural networks to automatically identify if an educational video is lively or not.

A learner can typically perceive or judge the liveliness\(^1\) of an educational video both through the visual and the auditory senses. A lecturer usually makes a video lively by using several visual actions such as hand movement, interactions with other objects (board/table/slides) and audio actions such as modulating voice intensity, varying speaking rate etc. In the proposed approach, both visual and audio information from an educational video are combined to automatically predict the liveliness of the video. Note that a given lecture can also be perceived as lively based on the contextual information (e.g., a historic anecdote) that the lecturer may intersperse within the technical content. We however don’t address this dimension of liveliness in this work\(^2\).

This paper is novel in three important aspects. First, the proposed approach is the first of its kind that combines audio and visual information to predict the liveliness in a video. Second, a convolutional neural network (CNN) is used to estimate the setup (e.g., lecturer sitting, standing, writing on a board etc.) of a video. Third, Long Short Term Memory (LSTM) based recurrent neural networks are trained to classify the liveliness of a video based on audio and visual features. The CNN output determines which of the audio and/or visual LSTM output should be combined for the liveliness prediction.

We observe that there is a lot of variation in what is being displayed in an educational video, e.g., slide/board, lecturer, both slide/board and lecturer, multiple video streams showing lecturer and slide etc. These different visual setups usually indicate to what degree the audio and the visual information should be combined for predicting liveliness. For example, when the video feed only displays the slide or the board, the visual features do not play a critical role in determining liveliness. However, when the video is focussed on

\(^1\)defined as “full of life and energy/active/animated” in dictionary

\(^2\) Note that the human labelers who provided the ground truth for our database [6] were explicitly asked to ignore this aspect while rating the videos
the lecturer, the hand gestures, body postures, body movements etc.
become critical, i.e., the visual component plays a significant role
in making a video lively. Hence, we first identify the setup of a
video using a CNN based classifier. Next, we depend on the setup,
we either use both audio and visual information or use only the
audio information from a video for training/testing of the LSTM
networks. We train two separate LSTM based classifiers, one each
for audio and visual modalities, which take a temporal sequence
of audio/visual features from a video clip as input and predict if
the clip is lively or not. Finally, audio/visual features from a test
video clip are forward-propagated through these LSTMs and their
outputs are combined to obtain the final liveliness label.

We perform experiments on the StyleX dataset [6], and compare
our approach with baselines that are based on visual, audio and
combined audio-visual features. The proposed approach shows re-
relative improvement of 7.6% and 1.9% with respect to a multimodal
dataset and a deep network baseline using only the audio modality
respectively.

2. RELATED WORK

In this section, we discuss the relevant prior art in deep learning
and multimodal public speaking analysis in videos.

Deep Learning: Recently deep neural networks have been exten-
sively used in computer vision, natural language processing and
speech processing. LSTM [7], a Recurrent Neural Network (RNN) [8]
architecture, has been extremely successful in temporal modelling
and classification tasks such as handwriting recognition [9], action
recognition [10], image and video captioning [11, 12, 13], speech
recognition [14, 15] and machine translation [16]. CNNs have also
been successfully used in many practical computer vision tasks
such as image classification [17], action recognition [18], object
detection [19, 20], semantic segmentation [21], object tracking [22]
etc.. In this work, we use CNNs for visual setup classification and
LSTMs for the temporal modelling of audio/visual features.

Multimodal Public Speaking Analysis: Due to the recent develop-
ment of advanced sensor technologies, there has been significa-
nt progress in the analysis of public speaking scenarios. The
proposed methods usually employ use of multiple modalities such
as microphone, RGB camera, depth sensor, kinect sensor, Google
glasses, body wearables, etc. and analyse the vocal behaviour, body
language, attention, eye contact, facial expression of the speakers
along with the engagement of the audiences [23, 24, 25, 26]. Gan
et al. [23] proposed baseline methods to do the quantification of
several above mentioned parameters by analysing the multi-sensor
data. Nguyen et al. [24] and Echeverria et al. [25] used kinect sen-
sors to recognize the bodily expressions, body posture, eye con-
tact of the speaker and thereby, providing feedback to the speaker.
Chen et al. [26] presented an automatic scoring model by using ba-
sic features for the assessment of public speaking skills. It must be
noted that all these works rely significantly on the sensor data cap-
tured during the presentation for their prediction task and hence,
they are not applicable to educational videos that are available on-
line. Moreover, all these approaches use shallow and hand-crafted
audio features along with the sensor data. On the contrary, our pro-
posed method uses deep learning based automatic feature extrac-
tion method for both audio and visual modalities from the video,
and predicts the liveliness.

To the best of authors’ knowledge, this is the first approach that
uses a deep multimodal approach for educational video analysis.

3. PROPOSED APPROACH

In this section, we describe the details of the proposed approach.
We begin with the description of how a given video is modeled as
a sequence of temporal events, followed by the visual setup clas-
sification algorithm. Next, we provide the details of the audio and
visual feature extraction. Finally, the details of the proposed mul-
timodal method for liveliness prediction is described. The pipeline
of the proposed approach is shown in Figure 1. The input to the
system is a fixed length video segment of 10 seconds during both
training and testing (referred to as 10-second clips throughout the
paper). For any educational video of arbitrary length, 10-second
clips are extracted with 50% overlap between the adjacent clips
and the overall video liveliness label is determined based on the
majority voting. In Section 5.1 we provide further details regard-
ing extraction of these 10-second clips from the Stylex dataset.

3.1 Video Temporal Sequencing

Each 10-second clip is modeled as a temporal sequence of smaller
chunks. If the total number of chunks in a 10-second clip is T,
then \( \{v_1, v_2, ..., v_{T} \} \) and \( \{a_1, a_2, ..., a_{T} \} \) represent
the temporal sequence of visual and audio features corresponding
to each 10-second clip respectively. Note that, \( v_t \) (Section 3.3) and
\( a_t \) (Section 3.4) are input to the visual and audio LSTM at time
instant \( t \).

3.2 Visual Setup Classification

One of our objectives is to automatically determine if both audio
and visual information are required for liveliness prediction. If a
video displays only slide/board, the visual features are less likely
to contribute to the liveliness. However, if the camera displays that
the lecturer is in a sitting/standing posture or is interacting with
the content, the visual features could significantly contribute to the
video liveliness. Hence, we collect a training dataset and train a
CNN to automatically estimate the setup of a video. We describe
the definition of the labels, the data collection procedure and the
details of the CNN training in the next three subsections.

3.2.1 Video Setup Label Definition

We define five different categories which cover almost all of the
visual setups usually found in educational videos.

- **Content**: This category includes the scenarios where the video
  feed mainly displays the content such as a blackboard or a slide
  or a paper. Frames, where the hand of the lecturer and/or pens or
  pointers are also visible, are included in this category. However,
  the video clips belonging to this category should not include any
  portion of the lecturer’s face. Since the lecturer is not visible in
  this case, only the audio modality will be used for liveliness
  prediction.
- **Person Walking/Standing**: In this scenario, the content such
  as blackboard/slide are not visible. However, the lecturer walks
  around or remain in a standing posture. The lecturer’s face and
  upper body parts (hand/shoulder) should be visible. Both audio
  and visual modality are used to predict liveliness in this case.
- **Person Sitting**: The content is not visible and the camera should
  focus only on the lecturer in a sitting posture. Both audio and
  visual modalities are considered for liveliness prediction.
- **Content & Person**: This includes all the scenarios where the up-
  per body of the lecturer and the content both are visible. Frames,
  where the lecturer points to the slide/board or writes something
  on the board, are included in this category. Here also both the
  modalities are used for liveliness.
• **Miscellaneous:** This category includes all other scenarios which are not covered in the above four categories, e.g., two different video feeds for professor and content, students are also visible, multiple people (laboratory setups) are visible in the scene etc. Since the frames from this category have significant intra-class variation and noise, we use only the audio information for liveliness prediction.

Some example frames from the above five categories are shown in Figure 2. The intra-class variation clearly shows the inherent difficulty of the setup classification task.

### 3.2.2 Label Collection

We used the StyleX dataset [6] for the liveliness prediction task. Although the liveliness labels were available along with the videos, video setup labels were not available. So we collect these additional labels using Amazon Mechanical Turk. We asked the Mturkers to look at the 10-second clips from StyleX and choose one of the five labels defined above. Each video clip is shown to three MTurk labellers and we assign the labels where at least two of the three labellers agreed. Although in most of the clips, all frames belong to only one of the above five categories, there were some 10-second clips (around 5%) where frames from more than one categories were present. In those cases, labellers were asked to provide the label based on the label of the majority of frames.

### 3.2.3 CNN for Label Classification

We used a CNN architecture to classify the setup of a 10-second clip. During training phase, all the frames belonging to a 10-second clip are used as the samples for the corresponding clip category. For this task, we use the same CNN architecture as used in [17]. In [17], the authors proposed a novel neural network model called Alexnet which improved the state-of-the-art imagenet classification [27] accuracy by a significant margin. Researchers in the computer vision community have often used the Alexnet architecture for other kinds of computer vision applications [28, 29]. Deep neural networks usually have millions of parameters. If the available training data for a particular classification task is not large enough, then training a deep neural network from scratch might lead to over fitting. Hence, it is a common practice to use a CNN which is already pre-trained for a related task and fine-tune only the top few layers of the network for the actual classification task.

We fine-tune the final three fully connected layers (fc6, fc7, fc8) of Alexnet for visual setup classification. First, we remove the 1000 node final layer fc8 (used to classify 1000 classes form imagenet [17]) from the network and add a layer with only five nodes because our objective is to classify each frame into one of the five setup categories. Since, the weights of this layer are learned from scratch we begin with a higher learning rate of 0.01 (same as Alexnet). We also fine tune the previous two fully connected layers (fc6 and fc7). However, their weights are not learned from scratch. We use a learning rate of 0.001 for these layers while performing the gradient descent with the setup classification training data. Once the Alexnet has been fine-tuned a new frame can be forward propagated through this network to find the classification label. For a test 10-second clip, we determine the setup label for each frame individually and assign the majority label to the full clip. We refer to this CNN as Setup-CNN.

### 3.3 Visual Feature Extraction

In this section, we describe the details of the visual features used for predicting the liveliness of a video clip. The visual modality is
Table:

<table>
<thead>
<tr>
<th>Labels</th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content (Only Audio)</td>
<td><img src="image1" alt="Example 1" /></td>
<td><img src="image2" alt="Example 2" /></td>
<td><img src="image3" alt="Example 3" /></td>
</tr>
<tr>
<td>Person Walking/Standing (Audio and Visual both)</td>
<td><img src="image1" alt="Content_01" /></td>
<td><img src="image2" alt="Content_02" /></td>
<td><img src="image3" alt="Content_03" /></td>
</tr>
<tr>
<td>Person Sitting (Audio and Visual both)</td>
<td><img src="image1" alt="Person_01" /></td>
<td><img src="image2" alt="Person_02" /></td>
<td><img src="image3" alt="Person_03" /></td>
</tr>
<tr>
<td>Content &amp; Person (Audio and Visual both)</td>
<td><img src="image1" alt="Content_01" /></td>
<td><img src="image2" alt="Content_02" /></td>
<td><img src="image3" alt="Content_03" /></td>
</tr>
<tr>
<td>Miscellaneous (Only Audio)</td>
<td><img src="image1" alt="Miscellaneous" /></td>
<td><img src="image2" alt="Miscellaneous" /></td>
<td><img src="image3" alt="Miscellaneous" /></td>
</tr>
</tbody>
</table>

Figure 2: Example frames from different visual setup categories. We also point out the modalities which are used for liveliness in each of these setups.

used to capture the movement of the lecturer. We used a state-of-the-art deep CNN architecture to represent the visual information in the form of motion across the frames. Unlike the CNN model used in Section 3.2.3 (where input to the network was an RGB image comprising of 3 channels), the input to the CNN model in this section is formed by stacking horizontal and vertical optical flow images from 10 consecutive frames of a video clip. We refer to this CNN model as Motion-CNN in the subsequent sections of the paper.

For the Motion-CNN, we fine-tuned the VGG-16 temporal-net trained on UCF-101 [30] action dataset. The final fully connected layers (fc6, fc7, and fc8) of VGG-16 are fine-tuned with respect to the liveliness labels of the videos. The activations of the fc7 layer are extracted as the visual representation of the stacked optical flows which were provided as the input to the model. Given a 10-second clip, we generate a feature representation $\eta_t$ (Section 3.1) from the corresponding 10 frame optical flow stack. We provide $\eta_t$ as an input to LSTM module at time $t$ to create a single visual representation for the full 10-second clip (Section 5.2).

Implementation Details: We use the GPU implementation of TVL1 optical flow algorithm [31]. We stack the optical flows in a 10-frame window of a video clip to receive a 20-channel optical flow image as an input (one horizontal channel and one vertical channel for each frame pair) to the Motion-CNN model. In Motion-CNN model, we also change the number of neurons in fc7 layer from 4096 to 512 before finetuning the model to get a lower dimensionality of the 10 frame optical flow stack. We adopt a dropout ratio of 0.8 and set the initial learning rate to 0.001 for fc6, and to 0.01 for fc7 and fc8 layers. The learning rate is reduced by a factor of 10 after every 3000 iterations.

3.4 Audio Feature Extraction

We extract the audio feature $\eta_t$ (Section 3.1) using a convolutional neural network. For each $t$, we find a corresponding one second long audio signal from the 10-second clip. We apply the Short-Time Fourier Transformation to convert each one second 1-d audio signal into a 2-D image (namely log-compressed mel-spectrograms with 128 components) with the horizontal axis and vertical axis being time-scale and frequency-scale respectively. The CNN features are extracted from these spectrogram images and used as inputs to the LSTM. We finetune the final three layers of Alexnet [17] to learn the spectrogram CNN features. We change the number of nodes in fc7 to 512 and use the fc7 representation corresponding to each spectrogram image as input to the LSTMs. The fine tuned Alexnet for the spectrogram feature extraction is referred as Audio-CNN. Learning rate and dropout parameters are chosen same as mentioned in Section 3.3.

3.5 Long Short Term Memory Networks

The Motion-CNN (Section 3.3) and the audio-CNN (Section 3.4) model only the short-term local motion and audio patterns in the video respectively. We further employ LSTMs to capture long-term temporal patterns/dependencies in the video. LSTMs map the arbitrary length sequential information of input data to output labels with multiple hidden units. Each of the units has built-in memory cell which controls the in-flow, out-flow, and accumulation of information over time with the help of several non-linear gate units. We provide a detailed description of LSTM networks below.

RNNs [8] are a special class of artificial neural networks, where cyclic connections are also allowed. These connections allow the networks to maintain a memory of the previous inputs, making them suitable for modeling sequential data. Given an input sequence $x$ of length $T$, the fixed length hidden state or memory of an RNN $h$ is given by

$$h_t = g(x_t, h_{t-1}) \quad t = 1, \ldots, T$$ (1)

We use $b_0 = 0$ in this work. Multiple such hidden layers can be stacked on top of each other, with $x_t$ in equation 1 replaced with the activation at time $t$ of the previous hidden layer, to obtain a ‘deep’ recurrent neural network. The output of the RNN at time $t$ is computed using the state of the last hidden layer at $t$ as

$$y_t = \theta(W_y h^n_t + b_y)$$ (2)

where $\theta$ is a non-linear operation such as sigmoid or hyperbolic tangent for binary classification or softmax for multiclass classification, $b_y$ is the bias term for the output layer and $n$ is the number of hidden layers in the architecture. The output of the RNN at desired time steps can then be used to compute the error and the network weights updated based on the gradients computed using Back-propagation Through Time (BPTT). In simple RNNs, the function $g$ is computed as a linear transformation of the input and previous hidden state, followed by an element wise non-linearity.

$$g(x_t, h_{t-1}) = \theta(W_h x_t + W_h h_{t-1} + b_h)$$ (3)

Such simple RNNs, however, suffer from the vanishing and exploding gradient problem [7]. To address this issue, a novel form of recurrent neural networks called the Long Short Term Memory (LSTM) networks were introduced in [7]. The key difference between simple RNNs and LSTMs is in the computation of $g$, which is done in the latter using a memory block. An LSTM memory
block consists of a memory cell $c$ and three multiplicative gates which regulate the state of the cell - forget gate $f$, input gate $i$ and output gate $o$. The memory cell encodes the knowledge of the inputs that have been observed up to that time step. The forget gate controls whether the old information should be retained or forgotten. The input gate regulates whether new information should be added to the cell state while the output gate controls which parts of the new cell state to output. The equations for the gates and cell updates at time $t$ are as follows:

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i)$$

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f)$$

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \phi(W_{cx}x_t + W_{ch}h_{t-1} + b_c)$$

$$h_t = o_t \odot c_t$$

where $\odot$ is the element-wise multiplication operation, $\sigma$ and $\phi$ are, respectively, the sigmoid and hyperbolic tangent functions, and $h_t$ is the output of the memory block. Like simple RNNs, LSTM networks can be made deep by stacking memory blocks. The output layer of the LSTM network can then be computed using equation 2.

We refer the reader to [7] for more technical details on LSTMs. The details of the architecture used in this work are described in section 5.2.

### 3.6 Multimodal LSTM for liveliness classification

In the proposed approach, LSTMs are used to learn the discriminative visual and audio feature representations for liveliness. The estimates from audio and visual LSTMs are combined to estimate the overall liveliness of videos. For setup categories ‘Person Walking/Standing’, ‘Person Sitting’ and ‘Content & Person’ setup, both the modalities are used for liveliness prediction. For the remaining videos from ‘Content’ and ‘Miscellaneous’ categories, only the audio LSTM representation is used to determine the liveliness label.

The details of the proposed approach are described below:

- **Visual-LSTM**: A multi-layer LSTM network is trained to learn the discriminative visual features for liveliness. The number of layers and the number of nodes in each layer in the LSTM network are determined based on a validation dataset. The input to the network at each time step $t$ is a 512 dimensional visual feature extracted as described in 3.3.

- **Audio-LSTM**: The approach for training an audio LSTM is similar to that for training the visual LSTM. The only difference is that the visual features are replaced by the audio features as described in 3.4.

- **Multimodal-LSTM**: Once we learn the discriminative audio and visual LSTMs, the next step is to combine their predictions to determine the final liveliness. The visual and audio features from each 10-second clip are now forward-propagated through the visual-LSTM and audio-LSTM respectively. Once the features corresponding to all the time-steps of a clip have been forward-propagated, the liveliness prediction from each of these LSTM networks are obtained. If the setup corresponding to a clip requires combining audio and visual modality information, we assign the clip a positive liveliness label if any one of the visual-LSTM or Audio-LSTM network predicts the label of the clip as positive. Otherwise, the audio-LSTM label is used as the final label for the 10-second clip.

The proposed multimodal pipeline for liveliness prediction is called LIVELINET and will be referred as that from now on.

### 4. BASELINE DETAILS

In this section, we describe several baselines which do not use any deep neural network for feature extraction or classification. However, these methods have demonstrated state-of-the-art accuracy in many many video/audio classification applications. We wanted to evaluate how good these “shallow” methods perform on the liveliness prediction task.

#### 4.1 Visual Baseline

The visual baseline consists of training a SVM classifier on state-of-the-art trajectory features aggregated into local descriptors. Improved Dense Trajectories (IDT) [32] have been shown to achieve state of the art results on a variety of action recognition benchmark datasets. Visual feature points on the visual frames are densely sampled and tracked across subsequent frames to obtain dense trajectories. Once the IDTs are computed, VLAD (Vector of Locally Aggregated Descriptors) encoding [33] is used to obtain a compact representation of the video. We set the number of clusters for VLAD encoding at 30 and obtain a 11880-dimensional representation for each video. SVM classifier with RBF kernel is used for the classification. We compare this visual baseline against the proposed approach.

#### 4.2 Audio Baselines

We compare LIVELINET with two different audio baselines; the first one uses bag of audio words and the second one uses Hidden Markov Models (HMM). The audio features are computed at a frame rate of 10 ms. The features are computed using the open source audio feature extraction software OpenSMILE [34]. Motivated by the findings in [35] and [36], where the authors show superior performance on various paralinguistic challenges, our frame-level features consist of (a) loudness, defined as normalized intensity raised to a power of 0.3, (b) 12 Mel Frequency Cepstral Coefficients (MFCCs) along with the log energy (MFCC0) and their first and second order delta values to capture the spectral variation, and (c) voicing related features such as the fundamental frequency (F0), voicing probability, harmonic noise ratio and zero crossing rate. (Intensity and fundamental frequency features have been found to be beneficial in liveliness classification in [4] also.) Authors in [36] refer to these frame-level features as Low Level Descriptors (LLD) and provide a set of 21 functionals based on quartile and percentile to generate chunk level features. We use all of these LLDs and the functionals for the audio feature extraction. For every one second audio signal (obtained using the same method as described in Section 3.4), these frame-level features are concatenated to form a $44 \times 100 = 4400$ dimensional feature vector. The dimensionality of the chunk-level audio feature is further reduced to 400 by performing a PCA across all the chunks in the training data.

The audio features from all the one second audio signals in the training videos are clustered into 256 clusters. A nearest neighbor cluster centre is found for each of these audio features. We then create a 256-dimensional histogram for each clip based on these nearest neighbour assignments. This approach, known as the bag-of-words model is popular in computer vision and natural language.
processing, and is beginning to be extended to the audio domain in the form of bag-of-audio-words (BoAW) (e.g., [37]). A SVM classifier with RBF kernel is trained on this BoAW representation.

As a second baseline, two 3-state HMMs, one each for the positive and the negative class, are trained using the sequence of audio features computed on these one second audio signals. Only left-to-right state transitions are permitted with a potential skip from the first state to the third state. Each state is modeled as 16-mixture Gaussian Mixture Model. The 44 frame-level LLD are the inputs to the HMM framework. The Scilearn implementation of HMM is used.

4.3 Multimodal baseline

For combining the audio and video modalities we employ a classifier stacking approach. Stacking involves learning an algorithm to combine the predictions of other classifiers. We first train two SVM classifiers on audio and video features separately. The features and kernels used here are the same as the individual audio and visual baselines described earlier. Subsequently, another SVM classifier (with RBF kernel) is trained on the predictions of the audio and video classifiers to make the final prediction. We compare this baseline against the proposed multimodal classifier.

5. EXPERIMENTAL RESULTS

In this section, we provide the details of the experimental results. First, we describe the StyleX dataset followed by the details of the proposed LSTM network architecture and setup classification results. Next, we provide the liveliness classification results using the proposed multimodal deep neural network method. Finally, we perform some preliminary quality analysis of the lively/not-lively videos.

5.1 Dataset

We use the StyleX dataset proposed in [6] for our experiments. StyleX comprises of 450 one-minute video snippets featuring 50 different instructors, 10 major topics in engineering and various accents of spoken English. Each video was annotated by multiple annotators for liveliness. The scores from all annotators (in the range 0 − 100, where 0 implies least lively and 100 implies most lively) corresponding to a particular video were averaged to obtain the mean liveliness score. The bimodal distribution of the mean liveliness scores were analyzed to estimate the threshold for binary label assignment (lively and not-lively). All videos with liveliness score above the threshold were assigned to the positive class whereas the remaining videos were assigned to the negative class. At a threshold of 54, we have 52% videos in the negative class (Thus, a simple majority-class classifier would lead to 52% classification accuracy). Out of the 450 StyleX videos, we randomly choose 00% for training, 20% for validation and 20% for testing while ensuring a proportional representation of both the classes in each subset. Since the proposed method takes 10-second clips as input during training and testing, we further split each one-minute video into 10-second clips bookended by silence, with a 50% overlap across adjacent clips. Each of these 10-second clips are assigned the same label as the actual one-minute videos and are treated as independent training instances. Likewise, during test, the 10-second clips are extracted from one-minute videos. The label is predicted for each 10-second clip and the label of the one-minute video is determined based on the majority vote.

5.2 LSTM Architecture Details

The parameters of the proposed visual-LSTM and audio-LSTM were selected using the validation set. The learning rate was initialized to $10^{-4}$ and decayed after every epoch. Dropout rate of 0.2 was used for the activations of the last hidden layer. We tried nine different combinations for the number of hidden layers (1, 2, 3) and number of units in each layer (128, 256, 512), for both visual and audio modalities. Visual-LSTM with 2 layers and 256 hidden units and audio-LSTM with 2 layers and 256 hidden units led to the optimal performance on the validation set.

5.3 Setup Classification

In this section, we report the visual setup classification results obtained using the framework proposed in Section 3.2. As discussed in Section 5.1, the number of video clips used is 2700 for the training phase and 900 each for the validation and testing phase (all clips are approximately 10 seconds long). The network is trained with all the frames (~ 300K) extracted from the training video clips. At the time of testing, a label is predicted for each of the frame in a 10-second clip and their majority is taken as the label of the full clip. We evaluate 5-way classification accuracy of the video clips into different visual setups. Our proposed CNN architecture achieves a classification accuracy of 86.08% for this task. However, we notice that for the task of liveliness prediction, we only require the classification of video clips into two different classes - (a) clips requiring only audio modality, and (b) clips requiring both audio and video modality for liveliness prediction. For this task of binary classification (’Content or Miscellaneous’ vs ‘Person Walking/Standing or Person Sitting or Content & Person’), our system achieves an accuracy of 93.74%. Based on the visual setup label of a clip, we use either both audio/visual or only audio modality for liveliness prediction.

5.4 Liveliness Classification

In this section, we present the performance of proposed multimodal deep neural network for liveliness prediction. Figure 3 depicts the results of our experiments. We obtain an accuracy of 70.6% with the Visual-LSTM, an absolute improvement of 6.2% over the visual baseline. The two audio baselines of HMM and BoAW methods lead to an accuracy of 60% and 63.3%, respectively. The Audio-LSTM setup leads to 75.0% accuracy, an increase of 11.7% over the best audio baseline. The proposed Multimodal-LSTM method (LIVELINET) achieves an accuracy of 76.5% compared to 71.1% obtained using the audio-visual baseline, an absolute improvement of 5.4% (relative improvement of 7.6%). We are also relatively 1.9% better than using only the audio-LSTM. The boost in accuracy when using both the modalities indicates that the information available from audio and visual modalities are complimentary and the proposed approach exploits it optimally.
5.5 Qualitative Analysis
We also perform qualitative analysis of the videos that are predicted lively/not-lively by LIVELINET. Our goal is to determine the general visual and audio patterns that make a video lively. These are the preliminary analysis of exemplar lively and exemplar non-lively lectures. We continue to perform a more systematic and in-depth qualitative analysis to understand two aspects: (a) patterns that the proposed classifier identifies as representative of lively and of not-lively, and (b) general audio-visual patterns that may have influenced the human labelers in assigning the 'lively or non-lively' label. One of the current directions for extending this work is to understand pedagogically-proven best practices of teaching and codify that knowledge in the form of features to be extracted and fed to the classifier. Some example frames from lively and not-lively videos as predicted by LIVELINET are shown in Figure 4. Some of our initial findings are: (a) Lecturers who alternate between making eye contact with the audience and looking at the content are perceived as more lively. (b) Similarly, voice modulations and moving around in the classroom (as opposed to sitting in place) and specific visual references (like pointing to written content) to synchronize with the spoken content seem to positively influence perceived liveliness.

6. CONCLUSION
We propose a novel method called LIVELINET that combines visual and audio information in a deep learning framework to predict liveliness in an educational video. First, we use a CNN architecture to determine the overall visual style of an educational video. Next, audio and visual LSTM deep neural networks are combined to estimate if a video is lively or not-lively. We performed experiments on the StyleX dataset and demonstrated significant improvement compared to the state-of-the-art methods. Future directions include incorporating text-based features for a content-based liveliness scoring. We also note that LIVELINET is going to be part of our e-learning platform TutorSpace.

References
Figure 4: Some example frames from videos predicted as lively and not-lively by our proposed method LIVELINET. The setup labels predicted by the proposed Setup-CNN approach are also shown.


Semantic Features of Math Problems: Relationships to Student Learning and Engagement

Stefan Slater
Ryan Baker
Jaclyn Ocumpaugh
Teachers College Columbia University
525 W. 120th St
New York, NY 10027
{slater.research, ryanshaunbaker, jlocumpaugh}@gmail.com

Paul Inventado
Peter Scupelli
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
{pinventado, scupelli}@cmu.edu

Neil Heffernan
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609
nth@wpi.edu

ABSTRACT
The creation of crowd-sourced content in learning systems is a powerful method for adapting learning systems to the needs of a range of teachers in a range of domains, but the quality of this content can vary. This study explores linguistic differences in teacher-created problem content in ASSISTments using a combination of discovery with models and correlation mining. Specifically, we find correlations between semantic features of mathematics problems and indicators of learning and engagement, suggesting promising areas for future work on problem design. We also discuss limitations of semantic tagging tools within mathematics domains and ways of addressing these limitations.

Keywords
Text mining, semantic analysis, problem features, engagement, learning, correlation mining, mathematics corpora

1. INTRODUCTION
As content is developed at scale for online learning systems, particularly systems that leverage content developed by large numbers of authors, it becomes important to distinguish between problems which are well-written and conducive to learning and those which are poorly worded or ineffective to understand. Crowd-sourced content, where content is authored by a broader community [21], is a powerful and scalable method of content creation, which can be used to quickly develop and deploy new content and curricula ([46], [17]).

For this reason, it is critical that an equally scalable method of analyzing problem quality be developed, to prevent learning platforms that leverage crowd-sourced content from becoming dominated by ineffective content. In other platforms such as Wikipedia the quality of crowd-sourced materials is improved through substantial coordination between contributors [20]. However, there is relatively little work evaluating crowd-sourced learning content at scale. In contrast with more traditional educational measurement (from tests), where determining items’ ability to discriminate student knowledge is a standard part of item analysis [11], there has been less attention to this problem for online learning systems. While some researchers have attempted to determine which hints are more effective [18], or which problems are associated with more learning [14], these efforts have focused on what, but not why, particular system features can impact student, limiting their degree of general use. A more theoretical approach was taken by [49] where a design space of over 70 features characterizing Cognitive Tutor lessons was distilled and correlated with an automated gaming the system detector. However, this work identified the characteristics of tutor lessons using hand-coding, a method that is infeasible for larger datasets, and was limited to the relatively narrow space of problems designed by professional educational developers.

An alternative method for the analysis of the design of content in large-scale educational systems is text mining. There is a considerable amount of small-scale research on linguistic features that impact reading in mathematical contexts [47], but as [16] point out, many of the traditional readability indices used to study language at scale are limited in the features they consider. As a result, many early studies did not find a relationship between readability and performance in mathematics word problems [48].

As more advanced linguistic tools have become available, large-scale investigations of mathematics language have become more fruitful. For example, [44] have used LIWC [37] and Coh-Metrix [15] to study the effects of linguistic properties of mathematics problems ([44], [45]). [45] found that third-person singular pronouns (e.g., he, she) are significantly associated with correct answers and fewer hint requests in Cognitive Tutor problems. They found positive correlations between the use of work-related terms and learning, and negative correlations between the use of terms related to social constructs and learning. These findings highlight the potential value of linguistic features for better understanding learning, as well as the need to explore a wider range of semantic categories in a broader range of mathematics content areas.

In this paper, we use a discovery with models approach, generating prediction labels from automated detectors of student learning and engagement that were developed for the ASSISTments online learning system ([2], [32]). We build on [46]’s approach of using text mining software and text elements, such as HTML tags and Unicode characters, to distill features
from a corpus of mathematics problems. We then use correlation mining approaches to identify links between these features and our labels of student engagement and learning as a means for determining which combinations of linguistic features are associated with particularly effective problems.

### 1.1 ASSISTments

The current study uses data collected from the ASSISTments system. ASSISTments is an online intelligent tutoring system used by over 50,000 students annually for middle-school mathematics. It provides both formative and summative assessment as well as extensive student support (assistance) and detailed teacher reports. It also facilitates research using randomized controlled trials (RCTs) that allow researchers to conduct studies without interfering with instructional time [17].

Within the system, students are assigned problem sets that may vary on several dimensions. Problem sets can be differentiated in terms of how problems are assigned: (a) In Complete All problem sets, problem order may be randomized; students must correctly answer all of the questions assigned and cannot advance to the next problem unless they have answered correctly. (b) In If-Then-Else problem sets, students must correctly answer a specified percentage of questions correctly (default is 50%) in order to pass, or else they may be given additional problems. (c) Finally, in Skill Builder problem sets, students must get 3 consecutive correct answers in order to pass, thus allowing students who show mastery to move on quickly to new assignments while providing struggling students with extended practice.

The purpose of the current study is to evaluate the semantic content of ASSISTments problems was analyzed with Wmatrix [39], a corpus analysis and comparison tool that

### 2. DATA & METHODS

In this paper, we analyze 179,908 problems within the ASSISTments system, most developed by teachers. We study these problems using the features of the problems themselves, in combination with data from the log files of 22,225 students who used ASSISTments during the 2012-13 school year. We applied models from previous research on engagement and learning to these students’ log files in order to determine how these constructs are associated with features of the design of the problems, developed through linguistic analysis and other data about the problems. In doing this, we excluded from consideration features that had been previously used within the learning and engagement models described below, to prevent overfitting.

### 2.1 Learning & Engagement Measures

Learning and engagement were assessed automatically, using detectors or models of these constructs.

#### 2.1.1 Student Learning

Student learning was assessed by fitting the moment-by-moment learning model to the data [2]. The moment-by-moment learning model (MBMLM) attempts to infer the specific effect of each learning opportunity on a student’s overall mastery. We used [2]’s look-ahead-two probabilistic approach, which assumes that learning can occur at multiple points along a student’s trajectory of learning a skill, rather than [43]’s approach which assumes a single moment of learning. We also choose this formulation because it explicitly analyzes future performance, allowing us to focus on cases where students perform better than expected after encountering a particular problem. Using the MBMLM allows us to isolate the average learning associated with specific problems within the data and compare these averages to other problems that either lack or have particular features of interest.

#### 2.1.2 Automated Detectors of Engagement

Detectors of student engagement were developed using data from in situ classroom observations, conducted by experts certified in the Baker Rodrigo Ocupampa Monitoring Protocol (BROMP 2.0). The protocol is enforced by HART, an Android application designed specifically for the BROMP and freely available for non-commercial research [33], which enforces the protocol while facilitating data collection.

Upon completion of the observations, data mining techniques were then employed to provide models of each construct that were cross-validated at the student level. In this paper, affective models developed for three different populations of students were applied, matching urban, suburban, and rural models to student data based on the location of their schools, in order to ensure population validity [32]. A detailed description of the features and algorithms used in these detectors is given in [32] and [34].

#### 2.1.3 Applying Across-Student Measures of Learning & Engagement to Individual Problems

In this paper, both the MBML and the engagement models were used as indicators of problem effectiveness. This section describes how these models were aggregated across the 179,908 problems and 22,225 students in this study. The formulation of the MBMLM in [2] is calculated once for each problem, at the time of the first attempt, and there is only one estimate per problem. Therefore, MBML was estimated for each student based on the sequence in which the problem was seen. Problem-level measures were then produced by averaging the MBML values across all students who saw a given problem.

The affective models were applied by segmenting the data at 20-second intervals (matching the original approach used to develop the detectors), and then applying each model to each segment. Confidence values for each detector was averaged twice at the problem level: first for each student (in order to avoid biasing the estimates in favor of the affect experienced by students who spent longer working the problem), then across all students who had seen that problem. This resulted in five measures per problem (average boredom, confusion, engaged concentration, frustration, and gaming), which we used, along with MBMLM outcomes, as our dependent variables.

### 2.2 Feature Engineering

A number of different design features may influence student learning and engagement. In this paper, we explore features of both the problem text and its meta-text. Specifically, we look at word counts, lexical category features generated by a semantic tagger, and features generated from the metadata connected to the problem, which provides us with a separate source of semantic data (e.g., the use of mathematical notation which would not be captured by a semantic tagger) as well as with information about its use of tables, images, formatting, bolded or emphasized text.

#### 2.2.1 Wmatrix Semantic Tags

The semantic content of ASSISTments problems was analyzed with Wmatrix [39], a corpus analysis and comparison tool that
parses text at a word and multi-word level. As of 2004, this included 42,300 single word entries and over 18,400 multi-word expressions [38]. Wmatrix has been used in a number of analyses, including work to tag and identify lexical patterns in ontology learning [13] and work to study how students self-explain when learning science content [12]. Its semantic tagger uses a semi-hierarchical structure where all known words and multi-word units are classified into one of 21 lexical fields, represented with letters by its tagging system. These lexical fields may (or may not) be further subdivided in up to three different levels, which are represented in what we will refer to as the base tag.

Figure 1. WMatrix tagging system.

Within the lexical tag, we will refer to the lexical field (alphabetical) and the 1st, 2nd, and 3rd order subfields (numeric) as the base tag. Additional information about antonyms (black vs. white), comparatives (better, worse, more confusing, etc.), superlatives (best, worst, most confusing, etc.), gender (masculine, feminine, and neuter), and anaphoric status (i.e., contextual reference), may or may not be appended to a base tag. Wmatrix documents 234 distinct base tags, and represents a large number of additional possible labels through appendices.

In the ASSISTmentss data, 442 distinct Wmatrix tags (base + appendices) were identified. These tags were most likely to fall under 7 lexical fields: General & Abstract Terms (A), Numbers & Measurement (N), Social Actions, States, & Processes (S), Psychological Actions, States, & Processes (X), Names & Grammatical Words (Z), Money & Commerce in Industry (I), and Time (T).

2.2.2 Accommodating Known Wmatrix Limitations
Although Wmatrix has been evaluated for its effectiveness in a range of genres, domains, and historical periods [38], semantic taggers can have a number of limitations when applied to highly specialized domains ([28], [24]; [36]; [30]; [27]). For example, research has shown that words which contain more than one unit of meaning create challenges for taggers that apply only one label per word [41]. As a result, semantic taggers which work specifically with scientific language have become an area of research interest ([1], [10]), but the language of mathematics has not yet been as prominent.

As such, features generated by Wmatrix must be carefully checked within this data set and may need to be supplemented by domain-specific tags. For example, we found several Wmatrix tags that erroneously tagged high-frequency items that appeared in ASSISTment’s instructions to students, including problems that instructed students to enter fractions in a specific format in order to receive credit or which told students that they had 3 attempts left. Wmatrix treated many of these words (e.g., enter and left) as an indication of physical movement (M1, as in entering a building or turning left). A few erroneous tags also appeared to result from the development of Wmatrix as a tool for British English. For instance, ASSISTment users, who are primarily American English speakers, wrote a number of problems involving a person named Randy, whose name was automatically (and erroneously) tagged as involving sexual content.

To mitigate this issue, significant correlations were carefully inspected individually. This approach has been found to be useful in previous studies where semantic taggers were applied to new domains [12]. While the large size of the ASSISTment corpus limits our ability to address this problem completely, thorough efforts were made to examine and understand relationships discovered through the use of Wmatrix. In instances where Wmatrix applied a tag involving the wrong sense of a word for the context in which it was used, we have specifically noted this difference and what sense of a word or words the tag is capturing within ASSISTments.

2.2.3 Math Symbols and Other Textual Metadata
In addition to generating features with Wmatrix, we also generated features based on the metadata of each problem. We were primarily concerned with identifying Unicode characters that are semantically meaningful in mathematics contexts. In the ASSISTments corpus, we labeled 68 symbols, such as those for integrals, mean, standard deviation, and exponents. These domain-specific symbols present unique challenges to the teaching and learning of mathematics [40], but are not detected by most lexical analysis tools, which have not generally been developed for mathematics domains. In addition, we identified 14 HTML tags that were used to format ASSISTments problems, including tags used for boldface, italics, paragraph structure, and images. Because many of these functions can also alter the semantics of a problem, we also generated features that reflect these uses of HTML in problem metadata. These features were generated by counting the number of times that each HTML code was used in a problem, in parallel to the application of the Wmatrix tags discussed in previous sections.

3. RESULTS
To explore the relationship between these problem features and the BROMP-trained measures of engagement and learning, we correlated each problem feature to each predicted variable. We selected Spearman’s ρ as our correlation coefficient because of its increased robustness when correlating non-normal data as compared to other parametric coefficients such as Pearson’s R [50]. Additionally, with such a high number of comparisons being conducted it was necessary to adjust our significance criterion to account for the possibility of tests being incorrectly identified as significant. The Benjamini and Hochberg post-hoc procedure [4] was used to control for these false discoveries. A table of results by dependent variable is presented in Table 1, which also provides the average confidence level for each detector as a baseline measure for this data.

<table>
<thead>
<tr>
<th>Outcome Measure</th>
<th>Avg</th>
<th>Conf.</th>
<th>Total</th>
<th>Sig w/</th>
<th>Sig w/</th>
<th>0.05</th>
<th>0.10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bored</td>
<td>0.16</td>
<td>118</td>
<td>16</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Engaged Concentration</td>
<td>0.46</td>
<td>251</td>
<td>62</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Confusion</td>
<td>0.03</td>
<td>285</td>
<td>60</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frustration</td>
<td>0.04</td>
<td>216</td>
<td>36</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gaming the System</td>
<td>0.02</td>
<td>257</td>
<td>43</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Of the possible 2730 correlations, 1127 (41.3%) were statistically significant after controlling for multiple comparisons using Benjamini & Hochberg’s post-hoc control. More features were
significantly correlated with confusion than any other outcome measure, but large numbers of features were also correlated with gaming the system, engaged concentration, frustration and MBML. Boredom was correlated with fewer features, overall, than either of the other outcome measures. These broad findings suggest the potential for finding semantic features that may help to provide templates for improving the design of word problems.

3.1 Features associated with all outcome measures
In the following sections, we examine the relationships between our features and the individual outcome measures, but in order to provide a broad summary of which types of features had the largest effects, the absolute value of Spearman $\rho$ was averaged across all six outcome measures for each feature in this study. Among the 64 features that were significantly correlated with all six outcomes, the 10 with the highest $\rho$ average (shown in Table 2) were drawn from 5 lexical fields: Grammatical Bin (Z), General Terms (A), Time (T), Speech Acts (Q), and Numbers & Measurement (N). One HTML tag («p», paragraph) was also significant.

Table 2. 10 largest correlated features by average sig. | $\rho$ |

| Tag | Avg $|\rho|$ | GB | General | Pronouns | Confidence | Concentration | Learning | Boredom | Gaming |
|-----|--------|-----|---------|----------|------------|--------------|-----------|---------|---------|
| Z5  | 0.116  | 0.193 | 0.086  | -0.165  | 0.084      | 0.305        | 0.060    |
| Z5+Z5mu | 0.104  | 0.114 | 0.034  | -0.040  | 0.135      | 0.362        | 0.140    |
| A12- | 0.101  | 0.114 | -0.027 | 0.030   | 0.086      | 0.153        | 0.198    |
| T3-  | 0.091  | 0.084 | -0.034 | 0.055   | 0.074      | 0.144        | 0.153    |
| Q2.2 | 0.080  | 0.043 | 0.083  | -0.162  | 0.068      | 0.071        | 0.051    |
| T1.1.2| 0.076  | 0.076 | -0.051 | 0.031   | 0.067      | 0.116        | 0.116    |
| «p» | 0.071  | 0.149 | 0.054  | -0.127  | 0.015      | 0.064        | 0.015    |
| N1   | 0.069  | 0.061 | 0.076  | -0.077  | 0.082      | 0.080        | 0.035    |
| A5.4+| 0.066  | -0.028| 0.059  | -0.130  | 0.074      | 0.038        | -0.069   |
| Z6   | 0.056  | 0.108 | 0.020  | -0.034  | -0.077    | -0.032       | 0.071    |

Spearman’s $\rho$ is also shown for individual outcome measures, allowing us to examine the effects of these features in greater detail. Table 2 shows that Wmatrix’s Speech Acts tag (Q2.2, e.g., answer, account, or speak out) is correlated with small increases in learning, but is also positively correlated with increased boredom and gaming and decreased concentration. The Wmatrix features described as Grammatical Bin (words such as as, but, in order to) are also correlated with increased learning, boredom, and gaming. Correspondingly, they are also negatively associated with engaged concentration, illustrating the complicated interactions at play in this data and the importance of considering multiple outcomes when exploring design effects.

4. Results by Outcome Measure
While some interactions are complicated, we also see many features correlate in logical patterns. For example, features that are positively associated with boredom are often also negatively associated with engaged concentration, and vice-versa. Likewise, features associated with confusion are also associated with frustration. The remainder of this section discusses these patterns in greater detail, pairing outcome measures that are conceptually related (e.g., boredom and engaged concentration as well as MBML and gaming the system, which have shown to be inversely related in the past). Specifically, we will examine the ten features that are most negatively associated and the ten that are most positively associated with each outcome measure, discussing commonalities across outcome measures.

4.1 Learning & Gaming the System
The Spearman $\rho$ values for the top ten features range from -0.078 to 0.233 for MBML and from -0.095 to 0.198 for gaming the system. Table 3 presents these results, highlighting features that correlate with both outcome measures.

Table 3. Features most strongly associated with MBML and gaming the system

<table>
<thead>
<tr>
<th>Tag</th>
<th>LEARNING</th>
<th>Tag</th>
<th>GAMING</th>
</tr>
</thead>
<tbody>
<tr>
<td>A17-</td>
<td>0.092</td>
<td>A10-</td>
<td>0.191</td>
</tr>
<tr>
<td>B1</td>
<td>0.133</td>
<td>M1</td>
<td>0.223</td>
</tr>
<tr>
<td>Z1-</td>
<td>0.149</td>
<td>S5</td>
<td>0.193</td>
</tr>
<tr>
<td>A5.2+</td>
<td>0.114</td>
<td>A5</td>
<td>0.105</td>
</tr>
<tr>
<td>Z5+Z5mu</td>
<td>0.104</td>
<td>M2</td>
<td>0.149</td>
</tr>
<tr>
<td>A5.1</td>
<td>0.104</td>
<td>Z5</td>
<td>0.132</td>
</tr>
<tr>
<td>A7+Z5uw</td>
<td>0.108</td>
<td>A2.1-</td>
<td>0.122</td>
</tr>
<tr>
<td>T2+</td>
<td>0.108</td>
<td>T1.1.2</td>
<td>0.121</td>
</tr>
<tr>
<td>A5.4-</td>
<td>0.086</td>
<td>T1.1</td>
<td>0.121</td>
</tr>
<tr>
<td>Z5mu</td>
<td>0.086</td>
<td>T1</td>
<td>0.121</td>
</tr>
</tbody>
</table>

Although gaming is an infrequent behavior, previous research has shown that it is linked to poorer learning [7, 34]. Therefore the findings in Table 3 are somewhat surprising. We should expect gaming’s influence to link overlap between the two categories, and expect them to show inverse relationships when present. Instead, A12- (words related to difficulty), Z5mu (multiword grammatical units like as far as or for example), and N3.8+ (words related to higher speeds), are all associated with increased MBML and increased gaming behaviors. Likewise, semantically similar categories like N1mu (multiword numbers) and N5+ (large quantities) are associated with lowered MBML and lowered rates of gaming behaviors.

These anomalies might be due to the existence of problems that support learning but can be gamed relatively easily, or might suggest that particularly challenging problems lead to learning but also inspire gaming behaviors. For example, A5.2+ (words associated with true) demonstrates the lowest correlation with learning, a result that is consistent with literature on the ineffectiveness of true/false questions [42]. Likewise Z5mu (multiword pronouns, e.g., anything at all) is correlated with lower MBML, while Z8 (single word pronouns, e.g., it, my, and you) is correlated with increased gaming. These findings align with research showing that pronouns can be difficult to process cognitively (taxing working memory), as they require readers to infer their antecedents (the words that give them their meaning) from context [25, 8, 22, 6]). This suggests that pronouns could inhibit learning by drawing mental resources away from mathematics task, perhaps inspiring some students to try to succeed with minimal cognitive effort.

These findings highlight important considerations for researchers working to improve learning systems, including the need to consider multiple measures. For example, [44] found that pronouns are associated with correct answers and lowered hint use. It is highly likely that pronouns can have beneficial impacts on learning, particularly through [44]’s hypothesized mechanism of increased cohesiveness. However, if pronoun use in ASSISTments and Cognitive Tutor is comparable, our results suggest that some correct answers could have been achieved by guessing rather than by learning.
Furthermore, if students are more tempted to game the system when presented with challenging problems, even though these are exactly the sort of problems needed to improve learning, then further research should explore whether or not these findings reflect two distinct different groups of students. It may be that some students need additional cognitive scaffolding or a motivational intervention in order to complete these problems without gaming, allowing them to learn as well as other students who are working through the curriculum in a more appropriate way. However, research has also shown that in some cases high achieving students also game the system, and the independent application of these models could be picking up on that trend, where students guess something that they actually know, but then correct this behavior in subsequent problems, which could cause the MBML model to perceive learning.

### 4.1.2 Confusion & Frustration

Confusion and frustration show considerable overlap, in line with prior theory on the relationship between these constructs ([19], [26]). As Table 4 shows, half (10) of the semantic features most strongly associated with one are also strongly associated with the other, including N6m (frequency of occurrence) which is negatively associated with both confusion and frustration. This corresponds with [44]’s findings that clear demarcations of time in mathematics problems can improve student outcomes.

**Table 4. Features most strongly associated with confusion and frustration**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Semantic Description</th>
<th>p</th>
<th>Tag</th>
<th>Semantic Description</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1.5.1</td>
<td>Using</td>
<td>-0.162</td>
<td>N6+</td>
<td>Quantities</td>
<td>-0.110</td>
</tr>
<tr>
<td>A1.5.1</td>
<td>Using</td>
<td>-0.097</td>
<td>N5+</td>
<td>Quantities</td>
<td>-0.070</td>
</tr>
<tr>
<td>N7.4</td>
<td>Measurement/Volume</td>
<td>-0.162</td>
<td>N1.1++</td>
<td>Important/Significant</td>
<td>0.063</td>
</tr>
<tr>
<td>N1.3</td>
<td>Measurement/Distance</td>
<td>-0.097</td>
<td>A11.8++</td>
<td>Important/Significant</td>
<td>-0.086</td>
</tr>
<tr>
<td>N6+</td>
<td>Frequency of occurrence</td>
<td>-0.097</td>
<td>A2.2</td>
<td>Cause/Connected</td>
<td>-0.056</td>
</tr>
<tr>
<td>A2.2</td>
<td>Cause/Connected</td>
<td>-0.097</td>
<td>N6+</td>
<td>Frequency of occurrence</td>
<td>-0.052</td>
</tr>
<tr>
<td>A1.3</td>
<td>Using</td>
<td>-0.072</td>
<td>A4.2</td>
<td>Means/Method</td>
<td>-0.051</td>
</tr>
<tr>
<td>N5+</td>
<td>Quantities</td>
<td>-0.070</td>
<td>A2.2</td>
<td>Cause/Connected</td>
<td>-0.051</td>
</tr>
<tr>
<td>I3.3</td>
<td>Money Price</td>
<td>-0.098</td>
<td>A12++</td>
<td>Change</td>
<td>-0.049</td>
</tr>
<tr>
<td>D4.1</td>
<td>General Apperance/Phys'l Properties</td>
<td>-0.098</td>
<td>&lt;span&gt;HTML font adjustment&lt;/span&gt;</td>
<td>-0.049</td>
<td></td>
</tr>
<tr>
<td>Q1.2m+</td>
<td>Paper documents &amp; writing</td>
<td>0.148</td>
<td>N1.1++</td>
<td>Important/Significant</td>
<td>-0.148</td>
</tr>
<tr>
<td>N1</td>
<td>Numbers</td>
<td>0.082</td>
<td>N2.6m+</td>
<td>Investigate/examine/test/search</td>
<td>0.092</td>
</tr>
<tr>
<td>D4.2</td>
<td>Work &amp; Employment: profession</td>
<td>0.083</td>
<td>&lt;span&gt;HTML font adjustment&lt;/span&gt;</td>
<td>0.092</td>
<td></td>
</tr>
<tr>
<td>Z5</td>
<td>Grammatical bin</td>
<td>0.084</td>
<td>N6+</td>
<td>Frequency of occurrence</td>
<td>0.093</td>
</tr>
<tr>
<td>A12</td>
<td>Easy/Difficult</td>
<td>0.084</td>
<td>Z5</td>
<td>Grammatical bin</td>
<td>0.105</td>
</tr>
<tr>
<td>&lt;span&gt;HTML font adjustment&lt;/span&gt;</td>
<td>0.084</td>
<td>I3.1</td>
<td>Work &amp; Employment: generally</td>
<td>0.124</td>
<td></td>
</tr>
<tr>
<td>I3.2</td>
<td>Work &amp; Employment: profession</td>
<td>0.083</td>
<td>T1.1</td>
<td>Time: Old, new and young; age</td>
<td>0.116</td>
</tr>
<tr>
<td>Z5</td>
<td>Grammatical bin</td>
<td>0.084</td>
<td>A12++</td>
<td>Change</td>
<td>0.131</td>
</tr>
<tr>
<td>X8+</td>
<td>Trying</td>
<td>0.135</td>
<td>A12++</td>
<td>Change</td>
<td>0.131</td>
</tr>
<tr>
<td>Z5m+</td>
<td>Grammatical bin</td>
<td>0.135</td>
<td>Z5m+</td>
<td>Grammatical bin</td>
<td>0.162</td>
</tr>
</tbody>
</table>

Notable semantic features within this pairing include Z5 and Z5m+ bin. Both capture what are known as grammatical bin, which includes prepositions (of, to, after, amid), conjunctions (and, or, but), certain adverbs (e.g., as, so, which, than, when), the infinitival maker (to & verb), determiners (e.g., a and the) and certain auxiliary verbs (e.g., do). Previous research has suggested that the highly specific style of scientific language increases the use of these parts of speech, especially in the sort of definitional contexts that we might find in many learning contexts [3], [29], for example, notes that students sometimes struggle with prepositions. In fact, this pattern is sometimes referred to as the **stylistic barrier hypothesis** [31], which suggests that differences between the language students use at home and the language used in the classroom may interfere with the learning process.

**HTML features that correlate with confusion and frustration**

Features associated with concreteness (N3.4, N3.3, A2.2, A1.5.1, N5+, I1.3, O4.1, T2++) correlate with lowered confusion and frustration, matching the literature on the concreteness effect, which shows that concrete words are not only processed faster than abstract words in many experimentally controlled studies [23], the two may operate in separate neurological pathways ([19], [5]). These findings are hypothesized to be an artifact of the word-to-word mapping system the brain uses to process language, where concrete words may have stronger ties to more basic concepts. Interestingly, [23] have found evidence for similar pathways for emotion words, which are acquired early and considered quite basic to the human experience. While several of the Wmatrix categories that might correspond with [23]’s account of emotion words do not appear in this list (E3, E4, X4.1), X2.1, described as thoughts/beliefs, has the strongest negative associations with both frustration and confusion.

Other features which correlate with increased confusion and frustration may reflect the sort of meta-instructions teachers use to support students working with complex mathematical problems. Consider, for example, the tags in the following examples:

1. **You Z5m+ must S6+ show A10+ your Z8 work I3.1.**
2. **You Z5m+ have A9+ three N1 attempts X8+**
3. **Often N6+ it Z8 helps S8+ to Z5 write_01.2 Q1.2 down Q1.2 [.i1.2.2 your Z8 work I3.1.**
4. **Keep A9+ trying X8+**
5. **Do X8+ [.i1.3.1 your X8+ [.i1.3.2 best X8+] [.i1.3.3**
6. **Do A1.1.1 the Z5 difficult A12- problems A12- first N4**

Several of these tags (as given in bold, above: I3.1 work; S6+ must; Z5 to, the; X8+ attempts, trying; A12- difficult; N6+ often) are correlated with increased confusion or frustration. This finding may reflect a preemptive scaffolding practice (e.g., teachers provide these additional instructions when students are working on problems that they have struggled with in the past). However, it is important to rule out other possibilities. For instance, such additional instructions could distract or annoy the students. More seriously, it could also have priming effects.

### 4.1.3 Engaged Concentration & Boredom

Like confusion and frustration, we see considerable overlap in the features correlated with engaged concentration and boredom. However, unlike confusion and frustration, these two outcome measures are negatively associated with one another. Six of the features most negatively associated with concentration (N5+, N3.6, Z5, Q2.2, A4.1, and A5.4+) are among those most positively associated with boredom. Likewise, four of those most positively associated with concentration (A2.1+mwu, A6.1+++, T3, and A5.2+) are negatively associated with boredom.
Interestingly, X2.1 (thoughts/beliefs) is not as closely related to boredom and engagement as it was to confusion and frustration, but two other features typically associated with language about humans show desirable associations with these two outcome measures. For instance, S5+c (groups & affiliation) is associated with increased engaged concentration, while X2 (mental actions/processes) is associated with lowered boredom. Likewise, A8, which tags words related to seem or appear (both mental processes typically ascribed to human subjects), also leads to lowered boredom.

These semantic features, along with several others that correlate with lowered boredom (T2++mwu time demarcations and M6/mwu location/direction) may also be indicators that problems with greater narrativity improve student engagement. However, we must still be cautious about interpreting lowered boredom as a desirable effect in and of itself, since A5.2+ (words associated with true) is also associated with lower boredom. This type of item is unlikely to bore students, since they can answer and pass it quickly. However, readers may recall that this feature is also correlated with lower learning, as one might expect based on previous research on True/False questions [42].

### 5. DISCUSSION AND CONCLUSIONS

Our analyses of the ASSISTments corpus complements previous research on the relationship between learning and the language of mathematics problems, but extends this line of inquiry by including educationally relevant behaviors and affective states as part of the learning outcomes measured. As discussed, a number of linguistic features (e.g., pronouns, mental states, time, and concreteness) have been found to be significant in previous work. However, we were also able to examine the degree to which these relationships reflect expectations about how behavior, affect, and learning are related.

For instance, some of the same features which were correlated with learning were also correlated with student frustration and gaming the system. While it might be hypothesized that frustrated students would be more likely to game the system, there is also evidence from within ASSISTments that frustration can be important for learning [26]. The MBML model used here is a look-ahead algorithm, which may optimize the opportunity to identify the problems that trigger learning even when learning process is causing student frustration. However, it’s also possible that these problems are triggering strong but distinct reactions in different students (e.g., students who persist vs. students who game the system when they become frustrated). Future work will hopefully shed more light on this unusual relationship.

Overall, these results point to a number of promising avenues for further research within the ASSISTments system. One key future approach will be to conduct RCTs of the features identified in this study, re-designing problems to eliminate problematic features or incorporate positive features, in order to determine whether our findings can drive enhanced design. At the same time, it will be important to explore some of the interactions that may exist between different combinations of linguistic features, or between linguistic features and other behaviors or actions within the tutor.

We also found several unusual patterns in our data, such as some features being associated with increases in both learning and with gaming the system. We believe this may be due to our dataset containing two different populations of students – those who are persistent in the face of challenging and difficult problems and those who are frustrated by these problems and attempt to game the system to avoid working through them. We hope to understand this relationship in greater detail through RCTs (as discussed below). Ultimately, we hope to use our findings to construct guidelines for teachers creating their own content in the system, which can be embedded directly into the authoring tools teachers use, providing useful feedback on their problem design.

### 5.1 Randomized Controlled Trials

Having found a set of features that are associated with differences in student engagement and learning, our next step will be to conduct a set of randomized controlled trials (RCTs) to test whether the effects we found are genuinely causal, and whether re-designing problems based on these findings can improve student outcomes. By determining which of these features are causal, we can expand scientific understanding of learning and engagement in online learning systems. By developing methods for concretely improving math problems, we can develop better guidelines and recommendations for the many instructors (and others) developing problems for the ASSISTments platforms. In the longer-term, we hope to make all of the problems in the ASSISTments platform engaging and educationally effective for each of the growing number of students who use ASSISTments to learn mathematics and other subjects.

### 5.2 Continued Feature Engineering

Another important area of future work will be to conduct further feature engineering, particularly in terms of text features specific to the language of mathematics. One of the shortcomings of the current study is that the language of mathematics is poorly modeled in existing tools. In addition to challenges cause by domain or context-specific uses of certain words, many semantic taggers rely on syntactic probabilities that may be difficult to capture when math problems are interspersed with text. Simply developing taggers that can identify embedded mathematics formulas (e.g., labeling ‘3+2’ as addition) could help to ameliorate this issue. We hope that, by developing more robust tools for the analysis of this particular corpus, we will be able to better predict and understand learning and engagement.

As research progresses, features derived from combinations of Wmatrix tags will also become important since many of the sub-categories within and across Wmatrix’s lexical fields may be semantically similar enough, or co-occur frequently enough, to warrant combining them within ASSISTments data. For example, Wmatrix treats deciding as separate from choosing, selecting, and picking, but this division may not be useful in mathematics learning corpora. Likewise, feature combinations may help to...
contextualize Wmatrix categories that are prone to incorrectly categorizing high-frequency words. For example, since many features in this study are highly correlated with M1, combinations involving this tag may be used to differentiate its use in instructions to students (e.g., “You have 3 attempts left”) from its use in physical descriptions related to geometry (e.g., “Jill turns left and walks 3 more miles.”).

5.3 Directions for Future Work
In this paper, we discovered relationships between semantic elements of text in the ASSISTments system and learning, affective, and behavioral student outcomes. In doing so, this work contributes to the emerging body of research studying the design of mathematics problems at scale.

Our findings show that a large number of semantically meaningful relationships exist, some of which correlate with a wide range of learner outcomes. These features provide insights that will help to develop guidelines for effective problem designs in ITSs. However, the existing suite of tools available for large scale textual analysis may not be optimal for tagging the specialized language of mathematics found in the ASSISTments system. Thus an additional area for future work includes the development of semantic taggers that are more appropriate for mathematics corpora. These efforts will help us to better understand how the linguistic properties of math problems influence student success at scale. In turn, by exploring potential relationships between persistence and student perceptions of challenge, we can work to design mathematics problems that are both more informative and more engaging.

6. ACKNOWLEDGMENTS
This research was supported by the National Science Foundation (NSFDRL 1252297). Any opinions and findings expressed are the authors’ and do not necessarily reflect the views of the NSF.

7. REFERENCES


An Ensemble Method to Predict Student Performance in an Online Math Learning Environment

Martin Stapel  
Department of Computer Science  
Humboldt University of Berlin  
Berlin, Germany  
martin.stapel@hu-berlin.de

Zhilin Zheng  
Department of Computer Science  
Humboldt University of Berlin  
Berlin, Germany  
zhilin.zheng@hu-berlin.de

Niels Pinkwart  
Department of Computer Science  
Humboldt University of Berlin  
Berlin, Germany  
niels.pinkwart@hu-berlin.de

ABSTRACT
The number of e-learning platforms and blended learning environments is continuously increasing and has sparked a lot of research around improvements of educational processes. Here, the ability to accurately predict student performance plays a vital role. Previous studies commonly focused on the construction of predictors tailored to a formal course. In this paper we relax this constraint, leveraging domain knowledge and combining a knowledge graph representation with activity scopes based on sets of didactically feasible learning objectives. Specialized scope classifiers are then combined to an ensemble to robustly predict student performance on learning objectives independently of the student’s individual learning setting. The final ensemble’s accuracy trumps any single classifier tested.

Keywords  
educational data mining, student performance prediction, ensemble methods, knowledge graph

1. INTRODUCTION
Performance prediction is one cornerstone of a fully personalized learning environment and also an important component of the efforts to deliver quality education. Higher education institutes, for example, are striving to incorporate predictive elements into their educational processes to better support students. Online systems like Massive Open Online Courses, Intelligent Tutoring Systems (ITs) and increasingly Learning Management Systems (LMSs) also look for methods to compensate the lack of face-to-face interactions with teachers and the resulting problems with student’s retention, completion, and graduation rates. Knowledge engineering and Educational Data Mining (EDM) methods and tools have helped to increasingly sharpen the models of student knowledge within these environments.

The foundations for performance prediction and student modeling were introduced more than four decades ago with Knowledge Tracing [1] and have since been constantly refined and extended to build diverse student models [3, 7, 17]. Such models are widely used in ITs to allow for adaptive and personalized behavior. Technological advancements and innovations enabled the development of more elaborate online learning environments that reduce learning costs [8] and overcome space and time limitations. Through the use of such systems, previously inaccessible data about student’s learning behaviors and their activities are now at hand. Analyzing student activities has become an important EDM task [2].

Data mining and machine learning approaches are often employed for the student performance prediction task since classification is one of the most frequently studied challenges by data mining and machine learning researchers. Such analyzes showed the ability to predict student’s performance [15, 25] and even their drop out [14] in a broad range of educational technology environments. Usually, such prediction efforts are centered around a rather formal course students have to follow, like a university course or a structured online-only course. In this paper, we focus on a learning technology system that deliberately refrains from such a course structure.

This math learning system – called bettermarks – offers its users, students and teachers alike, guidance without imposing a course on them. The learning platform supports different curricula as well as flexible teacher interventions and leads students to a particular learning objective at their pace. The learning objectives range from introductory knowledge to advanced concepts. For our work in this blended K-12 learning environment where students either work in a traditional school setting or on their own, we opted to focus on performance data for the prediction task. We combine measured performance data with a knowledge graph representation of the platform’s learning objectives, without the need for a strict course structure. Pursuing the prediction problem from this angle fully utilizes the math content organization and thereby directly connects extensive domain expertise and machine learning methods. The knowledge graph models how learning objectives are interconnected via pre-knowledge requirements. We use this graph to identify didactically feasible activity scopes. Based on those, special-
ized classifiers are trained and finally combined to predict student performance on a learning objective in an ensemble.

The remainder of this paper is organized as follows. In Section 2, we review how student modeling is approached in traditional ITSs and recent research on student performance prediction in different environments. Section 3 introduces the specific usage scenario of the bettermarks platform, its distinct characteristics, and the dataset. The following Section 4 describes our research method, including the generation of the classifier ensemble. Section 5 presents our findings and Section 6 concludes the paper with a discussion.

2. STATE OF THE ART
For Intelligent Tutoring Systems, student modeling is one major task which has been used for making assumptions about student’s latent attributes. It uses observations of student’s performance (e.g., correctness of given answers) or student’s actions (e.g., the time a student spent on an exercise) to estimate student’s hidden attributes, like knowledge, preferences or even motivational state. Which usually cannot be detected directly.

A well-established method for student modeling has been used in various fashions for more than 40 years now is called Knowledge Tracing (KT). This technique was pioneered by Atkinson [1] and substantially developed by Corbett and Anderson. Their variant is based on a 2-state dynamic Bayesian network [7]. The observed variable is the student performance, and the student knowledge is the latent one which is estimated. Regarding student performance, there are two additional parameters to account for accidental and careless mistakes (slip) and solving an exercise despite not knowing (guess). The set of parameters is completed with one for any prior knowledge a student might already have and one for her learning rate. This standard KT model is often used for its abilities to provide skill level diagnostics. In recent years, a range of extensions to Knowledge Tracing have been proposed to mitigate some of its shortcomings. A particularly noteworthy one is Baker et al.’s contextual guess and slip model [3]. Recently, Pardos and Heffernan proposed an extension to the standard model to incorporate item-level difficulty [17].

Besides KT, other approaches exist. A comparably new option is called Performance Factor Analysis (PFA) which was proposed by Pavlik et al. [19]. Their student modeling method uses a logistic regression model with a reconfigured version of Learning Factor Analysis [6] whose skill variable is replaced by one parameter per item (e.g., exercise, question, knowledge component) and the student variable is dropped entirely. The model estimates the individual item difficulty as well as effects of prior successes and failures for each skill. It predicts student performance based on item difficulty and prior performances. Comparative analyzes of KT’s and PFA’s performance showed that either of them appear to be suitable for student modeling [4, 10, 19].

In learning environments without such semantically rich data and a domain model, data mining, and machine learning approaches are often applied for the performance prediction task. The goals here remain mainly the same, with additional emphasis on early warning and drop out prediction. In general, student’s prior performances are used to train different machine learning models to predict future test or exam performance, similarly to PFA. However, not all environments provide access to performance data. The steadily growing number of LMSs, for instance, do not always collect such data. In such environments, one has to resort to data about student’s activities. Hu et al. developed an early warning system based on student’s usage of an LMS utilizing metadata captured while students interact with the system [12]. The studied dataset includes information like login counts, time spent logged in, and metadata concerning homework assignments and was gathered during two semesters of a fully online university course with 300 enrolled students. The course required students to attend online classes and watch videos in specific time periods. To build their early warning system, the authors generated three datasets to create different periods to study (4, 8 and 13 weeks) and applied three often used classification techniques, C4.5, CART, and logistic regression. Additionally, Hu et al. employed AdaBoost to achieve greater prediction accuracy which led to the best performing classifier constructed from AdaBoost and CART. This classifier achieved a prediction accuracy of at least 0.972 on each of the three datasets. A similar scenario, yet more open, was studied by Zacharias who investigated student performance related to online activities in an LMS, which was used as part of a blended learning university course [29]. 134 students were enrolled in this course for one semester. To account for student-teacher and student-student interactions which could not be observed, all of the captured online activities were treated equally while searching for significant correlations with the student’s final grades. Out of 29 variables, almost 50% were found to be important. A stepwise regression yielded a model with four variables which were used in a logistic analysis to discriminate between failing and not at risk students. An overall classification accuracy of 81.3% was achieved. Predicting student performance in a timely fashion as done by Koprinska et al. underscores the usefulness of performance data [13]. Their studied dataset included submission sets, assessment information, and engagement data from a discussion forum. All of the data was gathered from different online systems used in a blended university course. Koprinska et al. defined their classification problem as a three class problem and divided the 224 participating students into high-, average- and low-level students based on exam performance at the end of the course. To predict the exam result, they employed a decision tree classifier which achieved an accuracy score of 72.69% using the complete course data. Using just the data from the first half of the course led to an accuracy score of 66.52%. Here, almost half of the used features are performance related.

Our work uses a similar approach to predict student performance in a blended K-12 learning environment. The critical difference between other datasets used in previous research and ours is that students on the bettermarks platform neither attend nor follow a formal course. The system provides teachers and students with “math books” for a term’s curriculum. Since the learning platform is often used supplementary to traditional lessons in class, teachers make use of the learning material at their discretion. Likewise, students in a self-regulated learning setting might pick a couple of learning objectives or decide to work through a whole
book on their own. The resulting freedom for students and teachers introduces a huge amount of diversity in the user behavior and poses challenges for performance prediction algorithms. To fully capture student behavior and overcome the problem of fitting a single prediction model based on diverse data sources, Eesa and Ayad proposed a domain-specific decomposition of different (online) learning related aspects [9]. The final prediction would consequently consist of an ensemble of classifiers specialized on each aspect’s data. Hence, the resulting model should be more generalizable and flexible than models build on single courses. Building on this idea, we focused on learning objectives as the common data underlying every user’s interaction and decomposed the math content organization of the platform into different activity scopes. Classifiers trained on those scopes act as base classifiers for the developed ensemble which robustly predicts student performance independently of their learning situation.

The particularly chosen focus on exercises (or learning objectives, for that matter) in our research is a crucial distinction to prior ensemble-based prediction works. Student performance within an ITS as well as on a paper post-test was predicted by Baker et al. utilizing ensembles of different student models (including the previously discussed BKT and PFA). The achieved results let the authors conclude that ensembling appeared to be only slightly better [4]. Looking further into the previous results and concentrating exclusively on post-test predictions did not yield better prediction results over the best individual models [18]. Again, different student modeling approaches were combined to ensembles. Gowda et al. found that ensembles build on large enough datasets (about 15 times more data than used in the previous two studies) can very well yield superior prediction performance, even with similar models as a base [11].

3. THE USAGE SCENARIO
The bettermarks system is an online math learning platform with more than 100k interactive exercises, covering K-12 math curricula (grades 4-10) in English, Spanish, German and Dutch language. It is designed to be used in math classes at school without implying a formal course structure. Teachers can decide to teach math entirely with the system, supplement their lessons with related bettermarks content right in class, or assign exercises as homework. At any time, teachers can be aware of their student’s progress through detailed reports which present high-level performance aggregation as well as every single solution attempt. The system can also support and guide students working on their own in a self-regulated learning setting without additional teacher interventions. Each month, more than 100k students across Europe and America use bettermarks.

Besides offering detailed textbook-like explanations of math topics, the primary means of learning math on the bettermarks platform are math exercises. Exercises are grouped into exercise series. Each series helps students achieve a well defined and fine-grained learning objective. Examples of such learning objectives are “Calculate the surface area of a prism given the edge lengths and the height” or “Find the zeros of linear and quadratic functions.” These series are arranged into digital books based on curricular themes and didactical concepts without imposing any curriculum structure on the user. Each book is organized similarly to a printed math book with chapters and series of exercises within these chapters. Behind those books that are visible to teachers and students lies a knowledge graph (not visible to users). This graph describes how learning objectives relate to each other regarding required prior knowledge.

3.1 Knowledge Graph
The idea of a concept map was first introduced in the 1970s by Novak. In his later work, he used this framework to organize and connect already acquired knowledge with new knowledge [16]. The usefulness of maps related to the original ideas for learning and assessment in technology-based learning environments has already been shown [24, 27]. Building on these concepts, the underlying structure of the bettermarks content is called a knowledge graph. This graph is built by connecting nodes concerning their pre-knowledge requirements. Each of the graph nodes represents a learning objective – a particular skill a student reaches once she successfully finishes a series of exercises designed especially for this skill. These objectives include introductory/elementary skills as well as core knowledge and advanced skills. The direction of an edge indicates which node is defined as the required pre-knowledge for another node. A particular node might have more than one pre-knowledge node. The entire bettermarks knowledge graph contains more than 1,500 learning objectives in total. A small subset of them is shown in Figure 1. A digital math book on the bettermarks platform includes a number of these learning objectives. Usually, not all of them are directly (or indirectly) related.
3.2 Data
The analysis in this paper focuses on the particularly well-frequented book “Calculating Percents” from the German version of the bettermarks system. From this book’s learning objectives, we chose one with a relatively large amount of required pre-knowledge as a classification target. It is called “Calculate decreased and increased base values in context” and located close to the end of the book. The data was gathered during the entire year of 2015 and includes student’s activities on the bettermarks platform 40 days before their first attempt on the classification target. The 40 day period allows students in a school setting to reasonably work their way to this objective. In total, the dataset includes performance measurements of 566 students on 903 different learning objectives which are the results of 10,363 solution attempts by 6th - 10th-grade students from all over Germany. A student is free to repeat an exercise series as often as she wants. Since the system presents the student’s best solution attempt to a teacher first, we also used this result for each student and learning objective. Table 1 shows a randomly chosen sample of the entire dataset with results on three learning objectives (represented by identifiers). The results correspond to the ratio of correctly solved exercises in a series. It is evident that not all learning objectives have been addressed by the same amount of attempts. The last column shows the highest success rate on the classification target achieved by a student within 3 hours of starting the exercise series for the first time. We noticed that students employed different strategies involving repetitions while solving exercise series which makes the success rate achieved in the first attempt a bad indicator for the final result a student settles on by continuing with the next series. Therefore, the 3 hours allow students some time to repeat the exercise series and also account for the fact that students might have the classification target during their math lesson at school and want to repeat the exercise series again at home. These collected performance measurements are used as possible features in our classification models.

4. RESEARCH METHODOLOGY
Over the course of the following section, our research method is discussed in detail, we were guided by a two-fold research focus: (1) Can an ensemble of classifiers based on the decomposed math content organization accurately predict student performance? (2) Given the usage scenario, is this approach suitable for an “early prediction” setting? Since the bettermarks system offers its users lots of flexibility, an early prediction task is different from a formal course’s early prediction task. In our case, the early prediction challenge is not transferable to a subset of the course’s allocated time and exercises. Instead, we looked into students showing low usage rates over the examined period. In our case (and in contrast to online-only environments), a lack of activity does not imply that students did not attend a regular math lesson and progressed in school.

In a first step, the math content was decomposed into activity scopes relating to the classification target. A following pre-processing step used different aggregations to gain better insights into the available dataset. The primary concerns that governed this step refer to how much of the data is missing and if the classifiers can learn from roughly balanced classes created by the class split. The first question is also relevant regarding the number of actually achieved learning objectives by students within the different scopes since those directly translate into the initial feature sets. Afterwards, six different algorithms were evaluated on each scope as base classifiers for the ensemble. The process is described in the Ensemble Construction section which also discusses the imputation and standardization strategies we employed. Following the final model selection, the ensemble’s weights were optimized. This step also concluded the generation of the entire ensemble.

4.1 Activity Scopes
To reflect the flexibility the learning system offers its users, we defined three activity scopes and constructed specialized classifiers for them. All scopes center around a particular subset of the knowledge graph’s vertices and thus decompose the graph into relevant groups related to the classification target. The subgraph spun by the classification target’s vertex via the pre-knowledge relation serves as the binding element between the three scopes.

The first scope includes all learning objectives that are part of the classification target’s pre-knowledge in the knowledge graph. These are all vertices connected directly or indirectly to the classification target through pre-knowledge relation edges. In total, those are 35 different learning objectives for our chosen classification target “Calculate decreased and increased base values in context.”

The classification target is located in the math book “Calculating Percents”. This book with all of its learning objectives creates the second activity scope, the math book scope. Excluding the classification target itself, the set of potential features for this scope contains 24 learning objectives. Since the book was created with didactical considerations in mind, the math book’s learning objectives are arranged similarly to the knowledge graphs vertices. Still, this scope and the pre-knowledge scope share only five learning objectives.

The final scope includes student’s activities on learning objectives that are not part of the math book’s scope. All of these learning objectives are part of the knowledge graph as well, but those are located in other math books. Nevertheless, the resulting set was not partitioned any further by their books. This scope could share up to 30 learning objectives with the first scope but does not include any from the book’s scope. Those would be the learning objectives the pre-knowledge scope does not share with the book’s scope. The actual number depends entirely on the student’s activities during the examined period. With these defined scopes we attempted to model the different paths teachers and students might have taken to approach the classification target.

4.2 Pre-processing
In Germany, the bettermarks system is often used in math classes to supplement regular lessons. Therefore, it is not expected that students solve a vast amount of exercise series over the chosen 40 days. Figure 2 shows that the median of different exercise series per student is at 14.5 series with the 0.75 percentile at 23 series.

This result suggests that the amount of gathered performance measures per learning objective could be rather sparse

Proceedings of the 9th International Conference on Educational Data Mining 234
Table 1: Sample of user IDs with success rates on different learning objectives

<table>
<thead>
<tr>
<th>user_id</th>
<th>Learning objectives</th>
<th>classification_target</th>
</tr>
</thead>
<tbody>
<tr>
<td>369947</td>
<td>PruZiPruZiRFo.LOB04</td>
<td>0.333</td>
</tr>
<tr>
<td>92083</td>
<td>PruZiPruZiRDr.LOB06</td>
<td>0.333</td>
</tr>
<tr>
<td>5625246</td>
<td>ZUZUProp.LOB01</td>
<td>0.429</td>
</tr>
<tr>
<td>347284</td>
<td>PruZiPruZiRFo.LOB04</td>
<td>0.208</td>
</tr>
<tr>
<td>361389</td>
<td>ZUZUProp.LOB01</td>
<td>0.417</td>
</tr>
</tbody>
</table>

Figure 2: Students solve a rather small number of different series with the median at 14.5 series (indicated as green vertical line)

Figure 3: Data Sparsity

for the majority of series. In fact, 566 students worked on 903 different learning objectives with an average of almost 20 different series per student. Further examination reveals that only 22 learning objectives had up to 70% of the data missing. The data sparsity is illustrated in Figure 3. It is important to employ a suitable data imputation strategy and apply feature selection means during the construction of the different classifiers later to cope with this sparse dataset.

We decided to split the classes at a success rate of 0.75. One class is composed of students with success rates lower than 0.75, whereas the second one contains students with success rates of at least 0.75 which would translate to a separation of top performing students from all other students. This class split has the benefit of dealing with quite balanced classes. Figure 4 shows the median success rate at 0.76 (red) and our class split slightly left to it at 0.75 (green). The resulting spread is 45.6% to 54.4% between both classes.

Figure 4: Measured success rates at the classification target. The red line indicates class split at 0.75 and the green one the median success rate at 0.76

The dataset does not contain the entire set of pre-knowledge learning objectives. Out of 35 possible learning objectives, only data for 16 is present. One possible explanation is that pre-knowledge learning objectives are not always part of a single term’s curriculum (but available for teachers to choose from). Hence, it is not expected that students work their way through the entire pre-knowledge of a particular learning objective during a short period. All of the expected 24 book scope’s objectives are present in the dataset.

4.3 Ensemble Construction

An ensemble of classifiers blends predictions from multiple models with a two-fold goal: The first intent is to boost the overall prediction accuracy compared to a single classifier. The second benefit is a better generalizability due to different specialized classifiers. As a result, an ensemble can find solutions where a single prediction model would have difficulties. A key rationale is that an ensemble can select a set of hypotheses out of a much larger hypothesis space and combine their predictions into one [22].

For our purposes, we started with a set of well-known classification algorithms and used nested cross-validation to determine their performance. The algorithm with the highest average accuracy score in each scope is afterwards chosen for final model selection. The performance of the best model was evaluated on a hold-out dataset (30% of the entire data). Once the model selection took place, the weights for the ensemble were adjusted, again, with cross-validation and the final ensemble’s performance evaluated on the hold-out dataset. The following sections describe the whole process in detail.
Table 2: Average accuracy achieved in nested cross-validation for each tested algorithm and scope

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Book</th>
<th>Pre-knowledge</th>
<th>Outside</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree with AdaBoost</td>
<td>0.715</td>
<td>0.634</td>
<td>0.525</td>
</tr>
<tr>
<td>k-Nearest Neighbors</td>
<td>0.629</td>
<td>0.609</td>
<td>0.546</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>0.682</td>
<td>0.659</td>
<td>0.538</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>0.654</td>
<td>0.636</td>
<td>0.467</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.679</td>
<td>0.652</td>
<td>0.550</td>
</tr>
<tr>
<td>Stochastic Gradient Descent</td>
<td>0.624</td>
<td>0.594</td>
<td>0.525</td>
</tr>
</tbody>
</table>

4.3.1 Selecting Algorithms

A set of six commonly used classification algorithms were chosen as potential base models. The set consists of Random Forest, Decision Tree with AdaBoost, Logistic Regression, k-Nearest Neighbors, Stochastic Gradient Descent and a Naïve Bayes implementation. For each scope, a classification pipeline was created. To impute missing data we opted for filling missing values with the mean success rate of the particular feature. Tests with the median and the mode did not significantly influence later on achieved classification results. The data was robustly standardized by removing the median and scaling the data according to the Interquartile Range (IQR). Each pipeline used a scope-specific variance threshold on the imputed data as feature selection mechanism. The actual threshold is determined during model selection (0-60% of the feature’s variance). The purpose is to remove features that do not meet the set threshold. This applies to features with low variance due to rather uniform student activities as well as to features with large amounts of imputed data.

To get a conservative and thus fairly unbiased base estimate of each classifier’s performance [26], we used nested stratified cross-validation with 10 folds on the outside and 5 folds on the inside with randomized search [5] over the parameter space. Depending on the algorithm, the search space was limited to reasonable values such as restricting the number of trees in a forest. The search included 100 sets of candidate parameters. Table 2 shows the results for each classification algorithm and scope. The best performing algorithm is highlighted in each column.

4.3.2 Model selection and Ensemble construction

AdaBoost on Decision Tree for the math book scope, Logistic Regression for the pre-knowledge scope and Random Forest for the outside scope were picked for the final model selection. It was done by 10-fold cross-validation and a random search over 750 sets of candidate parameters. The best performing model of each scope was afterwards chosen and re-trained on the entire training set for the ensemble.

As before with the nested cross-validation results, the accuracy ranking over the three scopes stayed the same — the book scope’s classifier performed best (0.705) followed by the pre-knowledge scope’s classifier (0.682). With a prediction accuracy of 0.594, the baseline classifier scores below all other approaches. The constructed ensemble achieved the best prediction accuracy with 0.735.

To construct the ensemble we opted for a soft voting strategy rather than using hard voting. A soft voting strategy has the significant advantage of weighing the three scopes differently. The alternative would be to use a majority decision among the three classifiers where each classifier’s vote weights equally. Instead, the ensemble uses soft voting to classify students based on the argmax of the sums of each classifier’s predicted probabilities. To determine the weights to be associated with each classifier, we used random search with 10-fold cross-validation on 3k parameter sets. The emerged ensemble with tuned weights was then tested on the hold-out part of the dataset.

5. RESULTS

To assess the performance of each classifier as well as of the entire ensemble more thoroughly we also added a baseline classifier. This simple classifier always predicts the majority class. Table 3 shows each classifier’s prediction accuracy on the hold-out dataset.

Since the ensemble showed an improved accuracy on the test set, we investigated the remaining classification errors further. Table 4 displays the confusion matrix for the book scope’s classifier which is the best single-scope classifier. As a comparison, Table 5 shows the confusion matrix for the final ensemble. Out of the two, the latter made slightly more errors of type I. This is especially unfortunate because in our case, false positive errors translate to students incorrectly classified as top performers even though they could not reach the required success rate threshold. In our setting, errors of this type are arguably more expensive than classification errors of type II where a student would be wrongly classified as a low scoring student. If our prediction method would be used to trigger human interventions a teacher might determine rather quickly if a student is able to pass a test or not. However, if the system fails to notify the teacher in the first place, she might not at all be aware of a potential problem with the student’s performance. Thus, the problem would be revealed after the student has already failed.

Table 3: Prediction accuracy on the test set

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Prediction accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.594</td>
</tr>
<tr>
<td>Pre-knowledge scope</td>
<td>0.682</td>
</tr>
<tr>
<td>Book scope</td>
<td>0.705</td>
</tr>
<tr>
<td>Outside</td>
<td>0.647</td>
</tr>
<tr>
<td><strong>Ensemble</strong></td>
<td><strong>0.735</strong></td>
</tr>
</tbody>
</table>

Table 4: Book scope classifiers’s confusion matrix

<table>
<thead>
<tr>
<th></th>
<th>Other students</th>
<th>Top performers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other students</td>
<td>51</td>
<td>18</td>
</tr>
<tr>
<td>Top performers</td>
<td>32</td>
<td>69</td>
</tr>
</tbody>
</table>

\[1\] The pipeline facility, as well as the used algorithms’ implementations are part of scikit-learn [20].

\[2\] The IQR is the range between the 1st quartile (0.25 percentile) and the 3rd quartile (0.75 percentile)
Table 5: Ensemble’s confusion matrix

<table>
<thead>
<tr>
<th></th>
<th>Other students</th>
<th>Top performers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other students</td>
<td>50</td>
<td>19</td>
</tr>
<tr>
<td>Top performers</td>
<td>26</td>
<td>75</td>
</tr>
</tbody>
</table>

Figure 5: Ensemble’s accuracy gain over book scope’s classifier per quartile

Lastly, to assess the ensemble’s ability to accurately predict student performance in an early prediction task, the accuracy of the best single-scope classifier and the ensemble was compared based on quartiles of student’s number of solved exercise series. As described above, 50% of the students in our dataset solved up to 14.5 different exercise series in the examined period. To be used effectively in an early prediction setting, a suitable classifier needs to be able to accurately predict the right class with few data points. Figure 5 shows the accuracy difference between the book scope’s classifier and the entire ensemble for each quartile. In the first three quartiles the ensemble predicts more students correctly than the book scope’s classifier. These results lead to the conclusion that our approach has the potential be used in an early prediction setting.

6. DISCUSSION AND OUTLOOK

We investigated an approach that decomposes the math content structure underlying an online math learning platform, trains specialized classifiers on the resulting activity scopes and uses those classifiers in an ensemble to predict student performance on learning objectives. Students using this particular math learning platform achieve learning objectives without a formal course imposed on them which is quite different from course-centered online-only or blended learning environments. We showed that looking closer at the math exercises helped us build a robust classification model that can cope with student’s notably diverse behavior due to the lack of a strict course framework. Using the knowledge graph to decompose the content domain enabled the individual prediction models to better grasp nuances of student’s activities.

In general, the results suggest that our approach yields a robust performance prediction setup that can correctly classify 73.5% of the students in the dataset. This is an improvement over every other classification approach we tested in our study. Further examinations revealed that the ensemble also outperforms the best single-scope classifier in an early prediction or early warning setting. Students with lower levels of activity would benefit the most from our ensemble approach since it clearly improves the prediction accuracy for those students, as we have shown. However, the increased prediction accuracy came with a price: a slight increase in false positives where students are wrongly classified as top performing students. Especially in our area of research, false positive errors like this should be reduced as much as possible if we want to improve educational processes and make a lasting impact on every stakeholder.

Looking closer at the classification errors, we found that in 12 cases the three scope classifiers unanimously attributed the wrong class to a student. Hence, the ensemble was not able to predict the class for these students correctly either. The reason is a shortcoming of the ensemble’s soft voting strategy which cannot overturn matching predictions among its base classifiers. Rather than using a simple weighted ensemble, it is possible to use stacking and thus introduce a second stage classifier. This classifier takes the prediction results of the ensemble’s base classifiers and employs them as features to predict the final class. The whole concept is known as stacked generalization and exists in different flavors [28]. Gowda et al. have already shown the significant benefits of more sophisticated ensemble methods in a prediction task [11]. Additionally, a number of different ensemble generation methods can be utilized to achieve better diversity within the base classifiers [21]. Besides extending the final ensemble with stacking and exploring the resulting benefits, our future work will include more performance related data, like the number of attempts or the total time a student has spent on a particular exercise series. These efforts will go hand in hand with additional feature selection strategies, and dimensionality reduction means to capture more scope-related nuances of student’s performances.

We also plan to investigate whether student’s diverse sequences of learning objectives can be used to improve feature extraction and selection. Scheiter and Gerjets’ results regarding the order of presented problems and performance improvements point to a possible connection [23].

While some of the discussed extensions seem obvious, the most important challenge is to develop our approach into a strategy suitable for any learning objective in this scenario. Our current approach uses a narrow set of learning objectives and a specifically tailored ensemble. These constraints reduce the cold start problem but require a good strategy to cope with missing data, as we have described. Nevertheless, the ensemble cannot easily be repurposed at scale. Hence, investigating different strategies leading to a broadly applicable solution will be our primary focus.

References


Predicting Post-Test Performance from Online Student Behavior: A High School MOOC Case Study

Sabina Tomkins¹, Arti Ramesh², Lise Getoor¹

¹University of California, Santa Cruz
²University of Maryland, College Park
satomkin@ucsc.edu, artir@cs.umd.edu, getoor@soe.ucsc.edu

ABSTRACT
With the success and proliferation of Massive Open Online Courses (MOOCs) for college curricula, there is demand for adapting this modern mode of education for high school courses. Online and open courses have the potential to fill a much needed gap in high school curricula, especially in fields such as computer science, where there is shortage of trained teachers nationwide. In this paper, we analyze student post-test performance to determine the success of a high school computer science MOOC. We empirically characterize student success by using students’ performance on the Advanced Placement (AP) exam, which we treat as a post test. This post-test performance is more indicative of long-term learning than course performance, and allows us to model the extent to which students have internalized course material. Additionally, we analyze and compare the performance of a subset of students who received in-person coaching at their high school, to those students who took the course independently. This comparison provides better understanding of the role of a teacher in a student’s learning. We build a predictive machine learning model, and use it to identify the key factors contributing to the success of online high school courses. Our analysis demonstrates that high schoolers can thrive in MOOCs.

Keywords
online education, high school MOOCs, student learning

1. INTRODUCTION
Massive Open Online Courses (MOOCs) have emerged as a powerful mode of instruction, enabling access around the world to high quality education. Particularly for college curricula, MOOCs have become a popular education platform, offering a variety of courses across many disciplines. Now open online education is being deployed to high schools worldwide, exposing students to vast amounts of content, and new methods of learning. Even as the popularity of high school MOOCs increases, their efficacy is debated [8]. One challenge is that the large amount of self direction MOOCs require may be lacking in the average high school student.

To understand the applicability of the MOOC model to high schoolers, we analyze student behavior in a year-long high school MOOC on Advanced Placement (AP) Computer Science. This course is distinguished from traditional college-level MOOCs in several ways. First it is a year-long course, while college MOOCs average 8-10 weeks in duration. This provides ample opportunity to mine student interactions for an extended period of time. Secondly, while traditional MOOCs have no student-instructor interaction, the high school MOOC that we consider incorporates instructor intervention in the form of coaching and online forum instructor responses. Evaluating the effectiveness of this hybrid model allows us to investigate the effect of human instruction on high school students, a group which may particularly benefit from supervision.

Finally, we introduce a post test as a comprehensive assessment occurring after the termination of the course. A valid post test should assess students’ knowledge on critical course concepts, such that students’ course mastery is reflected in their post-test score. We treat the Advanced Placement (AP) exam as a post test and consider students’ performance on this test as being indicative of long term learning. Previous MOOC research evaluates students on course performance [4]. While course performance can be a good metric for evaluating student learning in the short term, post-test performance is a more informative metric for evaluating long-term mastery.

We propose and address the following research questions, aimed at evaluating the success of MOOCs at the high school level.

1. Can high school students learn from a MOOC, as evidenced here by their post-test (AP exam) performance?
2. How does coaching help students achieve better course performance and learning?
3. How can we predict student’s post test performance from course performance, forum data, and learning environment?

Our contributions in this paper are as follows:

1. We perform an in-depth analysis of student participation and performance to evaluate the success of MOOCs at the high school level. To do so, we identify two course success measures: 1) course performance scores, and 2) post-test performance scores.
2. We evaluate the effect of two important elements of this high school MOOC: discussion forums and coaching, on student performance.

3. We use a machine learning model to predict student post test scores. First constructing features drawn from our analysis of student activities, then determining the relative predictive power of these features. We show that this process can be used to draw useful insights about student learning.

2. RELATED WORK

Research on online student engagement and learning, is extensive and still growing Kizilcec et al. [5], Anderson et al. [1], and Ramesh et al. [11] develop models for understanding student engagement in online courses. Tucker et al. [13] mine text data in forums and examine their effects on student performance and learning outcomes. Vigentini and Clayphan [14] analyze the effects of course design and teaching effect on students’ pace through online courses. They conclude that both the course design and the mode of teaching influence the way in which students progress through and complete the course. Simon et al. [12] analyze the impact of peer instruction in student learning.

Particularly relevant to our findings is the impact of gaming the system on long-term learning. Baker et al. [2] investigate the effect of students gaming an intelligent tutor system on post-test performance. In the high school MOOC setting, we observe a similar behavior in some students achieving high course performance, but low post-test performance. We identify plausible ways in which these students can be gaming the system to achieve high course performance and present analysis that is potentially useful for MOOC designers to prevent this behavior.

There is limited work on analyzing student behavior in high school MOOCs. Kurhila and Vihavainen [6] analyze Finnish high school students’ behavior in a computer science MOOC to understand whether MOOCs can be used to supplement traditional classroom education. Najafi et al. [9] perform a study on 29 participating students by splitting them into two groups: one group participating only in the MOOC, and another group is a blended-MOOC that has some instructor interactions in addition to the MOOC. The report that students in the blended group showed more persistence in the course, but there was no statistically significant difference between the groups’ performance in a post-test. In our work, we focus on empirically analyzing different elements of a high school MOOC that contribute to student learning in an online setting. We use post-test scores to capture student learning in the course and examine the interaction of different modes of course participation with post-test performance. Our analysis reveals course design insights which are helpful to MOOC educators.

3. DATA

This data is from a two-semester high school Computer Science MOOC, offered by a for-profit education company. The course prepares students for College Board’s Advanced Placement Computer Science A exam and is equivalent to a semester long college introductory course on computer science. In this work, we consider data from the 2014-2015 school year for which 5692 students were enrolled.

The course is structured by terms, units, and lessons. Lessons provide instruction on a single topic, and consist of video lectures and activities. The lessons progress in difficulty beginning with printing output in Java, and ending with designing algorithms. Each lesson is accompanied with activities. These activities are not graded, instead students receive credit for attempting them. Students take assessments in three forms: assignments, quizzes, and exams, each released every two weeks.

At the end of the year students take an Advanced Placement (AP) exam. Students can use their AP exam performance exam as a substitution for a single introductory college course. The AP exam score ranges from 1 to 5. In all, we have data for 1613 students who take the AP exam. This number is a lower limit on the total number of students who may have taken the course and the AP. The course provides a forum service for students, which is staffed with paid course instructors. Approximately, 30% of all students who created course accounts also created forum accounts, 1728 students in all.

This course is unique in that it provides a coach service which high schools can purchase. This option requires that the school appoint a coach, who is responsible for overseeing the students at their school. The coach is provided with additional offline resources, and has access to a forum exclusive to coaches and course instructors. The average classroom size is approximately 9 students with a standard deviation of approximately 12 students. The largest classroom size coached by a single coach is 72, while some coaches supervise a single student. Of all students who have enrolled in the course, approximately, 23% (1290) are coached and 77% (4402) are independent. From here on we refer to the students enrolled with a coach as coached students.

We summarize the class statistics in Figure 1 below. The majority of coached students sign up for the student forum, and many persist with the course to take the final AP exam at the end of the year.

![Figure 1: Student participation varies between coached and independent students.](image)

4. EMPIRICALLY CHARACTERIZING SUCCESS OF A HIGH-SCHOOL MOOC

In this section, we use post-test performance and course performance to question the success of MOOCs for high school...
students. With an empirical analysis, we provide insights on how to adapt high school MOOCs to benefit different groups of students. To investigate this question, we focus on the subset of students for whom we have post-test data. To evaluate student success in the course, we identify three measures of course participation in MOOCs that are relevant to the high school population: overall score, course completion, and post-test score.

**Overall Score** The overall score captures the combined score across course assignments, quizzes, exams, and activities, each of which contributes to the final score with some weight. We maintain the same weights as those assigned by the course, exams are weighted most heavily, activities the least.

\[
\text{Overall Score} = 0.3 \times (\text{Assignment Score} + \text{Quiz Score}) + 0.6 \times \text{Exam Score} + 0.1 \times \text{Activity Score}.
\]

**Course Completion** The second success measure we use is course completion. Course completion measures the total number of course activities and assessments completed by the student.

\[
\text{Course Completion} = \frac{\text{Total Activities and Assessments Attempted}}{\text{Total Number of Activities and Assessments}}
\]

**Post-Test Score** This score captures student scores in the post test that is conducted 2 weeks after the end of the course. The score ranges from 1 to 5. This score captures the advance placement (AP) score, hence we also refer to it as the AP score.

To evaluate the effectiveness of the high school MOOC on student performance, we first examine the relationship between course completion and course performance. We hypothesize that as students complete a higher percentage of the course, they should do better in the course assessments leading to higher course performance scores and post-test scores. Examining the correlation of course completion to post-test performance, we find that they are positively correlated. This suggests that the course indeed helps students in achieving good performance in the assessments. However, we find that of the students that achieve an overall score of 90 or greater, only 70% pass the post test. Similarly, of the students who complete 90% of the course, only 63% pass the post test. These initial observations indicate the need to perform a more detailed study in order to understand the different student populations in the course.

Next, we examine the relationship between overall score and post-test score, captured in Figure 2. From this plot, we see a positive linear relationship between course performance and post-test score. Notably, we observe that the average post-test score of the students who achieve an overall score of 4.0 or higher, and well above a passing score.

In Figure 4, we present results of student performance across assessments. Figures 4(a), 4(b), and 4(c) present average student assignment, quiz, and exam scores for students who passed/failed the post test, respectively. We find that students who pass the post test do better on assessments. We also observe that the scores across all assessments show a decreasing trend as the course progresses. This signals that the assessments get harder for both groups of students as the course progresses. Another important observation is the increase in scores for both groups at assignment 8, quiz 5, and exam 4; these assessments are at the start of the second term in the course, indicating that students may have higher motivation at the start of a term.

![Figure 2: The dot sizes are proportional to the number of students achieving the overall score.](image)

**Figure 3:** Students who pass are more likely to attempt assignments than students who fail.

Additionally, some assessments show a greater difference between the two groups of students, and performance on these assessments are more informative of student learning. In Figure 4(c), we observe that for both passed and failed students, we see the greatest dip in performance in the final exam. As the final exam is the most comprehensive exam, and possibly most related to the post test, analyzing why students do so poorly on this exam is a worthwhile direction of study in its own right.

Another important dimension is considering assignment com-
5. FORUM PARTICIPATION AND POST-TEST PERFORMANCE

In this section, we analyze forum participation of students and examine its effect on course success. To do so, we answer the following questions:

- Does participation in forums impact post-test performance and learning?
- What are the key differences between participation styles of students who pass the course and students who do not?

We first look at the average score of students who use the forum compared to the average score of students who do not use the forum. Students who use the forum have a statistically higher post test performance score of 2.77, whereas students who do not use the forum obtain a score of 2.34, (p < .001). It is not clear if the forum impacts learning, or if instead, students with a high desire to learn are more likely to use the forum.

To accurately evaluate forum participation of the two sub-populations, we analyze them on different types of forum participation. Forum participation comprises of different types of student interactions: asking questions, answering other student questions, viewing posts, and contributing to conversation threads. Table 1 gives the comparison of students who pass the post test against student who do not across the various forum participation types. The different types of forum participation types are referred to as: Questions, Answers, Post Views, and Contributions. We also consider the number of days that a student was logged into the forum, which is denoted by Days Online.

On average, students who pass the course make more contributions than students failing in the course. They also answer more questions. Both groups seem to spend roughly the same amount of time online, to view the same number of posts, and to ask the same number of questions. What most distinguishes a student who passes, from one who fails is whether they are answering questions and contributing to conversations.

This analysis further demonstrates the importance of forums to MOOCs. Answering questions and contributing to conversations are two behaviors indicative of strong post-test performance. We hope that MOOC designers can use this information to create appropriate intervention and incentive strategies for students.

6. COACHING

In this section, we evaluate the effect of coaching on student learning. We compare coached students to independent students using their participation in course assessments and forums. We conclude this section by looking at the subset of students who have only one coach, in order to isolate the effect of coaching from other classroom effects.

6.1 Course Behavior

![Average Assignment Score](image)

(a) Average assignment scores of passed and failed students

![Average Quiz Score](image)

(b) Average quiz scores of passed and failed students

![Average Exam Score](image)

(c) Average exam scores of passed and failed students

Figure 4: Passed students have higher average scores across all assessments than failed students.

We examine the relationship between attempting assignments and course performance and find that students passing the post test also attempt more assignments. This implies that the high scores of these students are not only the product of strong prior knowledge, but are also the result of learning from the course.

Table 1: The average forum participation is significantly more for students that pass the course. The behavior for which there was a statistical significance difference between the groups are highlighted in bold.
Average Assignment Score
Coached
Independent

Average Quiz Score
Coached
Independent

Average Exam Score
Coached
Independent

Figure 5: Coached students have higher average scores than independent students.

We inspect the average assessment scores of coached and independent students in Figure 5. Observing scores across assignments, quizzes, and exams in Figures 5(a), 5(b), and 5(c), respectively, we find that coached students perform better than independent students across all assessments.

Such differentially high performance in the course should indicate higher performance in the AP exam for coached students. However, we see that coached students fail to get a high post-test score. The average post-test score for a coached student is 2.43, while it is 2.59 for an independent student. We test statistical significance using a t-test with a rejection threshold of \( p < 0.05 \). In Section 6.2, we analyze forum participation of students to understand this difference in scores.

6.2 Forum Participation of Coached and Independent Students

Analyzing forum participation of coached and independent students, we find that there is a significant difference in forum participation between coached and independent students. Table 2 gives the comparison between coached and independent students in forum participation. On average, coached students ask more questions and answer fewer questions on the forums when compared to independent students. Coached students exhibit more passive behavior by predominantly viewing posts rather than writing posts, when compared to independent students. This can be particularly dangerous if the posts which are viewed contain assignment code.

In Table 3, we compare coached students who pass to coached students who fail and see the same differences as those observed between all students who pass, and all students who fail. Students who pass are more likely to answer questions, and contribute to conversations.

<table>
<thead>
<tr>
<th>Forum Behavior</th>
<th>Coached Mean</th>
<th>Independent Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Questions</td>
<td>2.81</td>
<td>1.90</td>
</tr>
<tr>
<td>Answers</td>
<td>1.45</td>
<td>1.72</td>
</tr>
<tr>
<td>Post Views</td>
<td>145.49</td>
<td>81.50</td>
</tr>
<tr>
<td>Contributions</td>
<td>8.10</td>
<td>7.33</td>
</tr>
<tr>
<td>Days Online</td>
<td>20.64</td>
<td>12.55</td>
</tr>
</tbody>
</table>

Table 2: Coached students view more posts and ask more questions. The behavior for which there was a statistical significance difference between the groups are highlighted in bold.

Table 3: The differences in forum behavior between coached students who pass and who fail follow the same trends in forum behavior exhibited by the general population, and shown in Section 5. The behavioral features for which there was a statistical significance difference between the groups are highlighted in bold.

6.3 Coaches with Only One Student

To examine the effect of coaching class size on coached students’ post-test performance, we examine coached students in a classroom size of one. Comparing average post test scores of coached students who are singly advised by their coaches (classroom size of one) with independent students, we find that the average post-test score for the coached students is 3.6, while it is 3.2 for independent students. We hypothesize that the lower score of coached students in classroom size greater than one is due to the possibility of sharing answers when students study together. This explains their high overall score but lower post-test scores. This analysis further suggests that the effect of coaching is confounded by the effects of learning in a classroom with peers. To fully
understand the effect of a coach guiding a student through the learning process, the peer-effects of classmates should be better understood and isolated. In Section 7, we take first steps in this direction by proposing student types.

7. INSPECTING UNEXPECTED STUDENT TYPES

In this section, we identify and analyze various types of students in the course based on their performance in the assessments. We classify students into two broad types based on whether the overall scores and post-test scores are correlated. Figure 6 gives the relationship between overall score and post test score for all students. Two groups of students emerge, students who exhibit a correlation between overall scores and post test scores, and students who do not. These two groups can be further broken down based on whether they obtain a high score on the post test, yielding four groups of students.

- **Low learners**: These students have low values for both overall scores and post test scores.
- **High learners**: These students obtain high values for both overall scores and post test scores.
- **Unexpected low learners**: These students obtain high overall scores, but low post test scores.
- **Unexpected high learners**: These students obtain high post test scores, but low overall scores.

Among these, the unexpected low learners and unexpected high learners deviate from the rest of the students. To analyze these two groups, we delve deeper into other aspects of the course such as forum participation and coaching.

![Figure 6: Four groups of students emerge: low learners, high learners, unexpected low and high learners. For high course performance we choose a threshold of 60% as a passing grade.](image)

**7.1 Unexpected Low Learners**

Unexpected low learners are those students who perform well on the course assessments (with an overall score of over 60%) but who do not earn a passing post-test score. We hypothesize that this might be due to their not retaining information from the course, or not arriving at high overall course scores on their own. To understand their low post-test performance, we examine their forum behavior and coaching environment.

As can be seen in Figure 7, approximately 91% of unexpected low learners are coached students. Most of these students are part of large classrooms coached by the same coach, increasing the possibility of getting answers from their peers/coach. Plagiarism is a significant challenge in online courses as proctoring students online is not as efficient as in classroom courses.

Further, analyzing forum performance, we find that approximately 76% of unexpected low learners use the forum. Of those who use the forum, 91% are coached. Table 4 gives the forum participation of coached and independent unexpected low learners. The forum participation of these students have a strong similarity to failing students in Table 1, participating passively in the course by viewing forum posts and contributing to less answers. The coached students are less active than the independent students on the forum in every way, even in post views. While it was posited before that active forum participation is indicative of learning and high AP exam performance, this may not be the case in all groups. For example, the small number of independent students may be using the forum for social, rather than learning purposes.

![Figure 7: The majority of unexpected low learners are coached, while the majority of unexpected high learners are independent.](image)

**Table 4: Forum behaviors for which there is a statistical significance between groups are highlighted in bold.**


<table>
<thead>
<tr>
<th>Forum Behavior</th>
<th>Coached Mean</th>
<th>Independent Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Questions</td>
<td>3.5</td>
<td>9.2</td>
</tr>
<tr>
<td>Answers</td>
<td>0.5</td>
<td>15.0</td>
</tr>
<tr>
<td>Post views</td>
<td>195.0</td>
<td>293.0</td>
</tr>
<tr>
<td>Contributions</td>
<td>7.1</td>
<td>67.0</td>
</tr>
<tr>
<td>Days Online</td>
<td>25.6</td>
<td>35.2</td>
</tr>
</tbody>
</table>

**7.2 Unexpected High Learners**

Unexpected high performers earn an overall course score of less than 60% but pass the AP exam with a 3 or above. Approximately 86% (357 out of 409) of unexpected high learners are independent and approximately 86% of the unexpected high learners (323 out of 409) are not on the forums. That this group can do so well on the post test, without either a high amount of course or forum participation strongly suggests that either these students have prior knowledge in computer science or that they are not being primarily exposed to
computer science through this course but are instead using it to supplement another mode of instruction. A pre test of students’ prior computer science knowledge would provide further clarity.

8. PREDICTING PERFORMANCE FROM STUDENT BEHAVIOR

In Sections 4 and 5, we see that students’ post-test performance is affected by their course and forum behavior. We construct features with which to model these different characteristics of student behavior. These student models are then used to predict post-test scores. By discovering the relative rank of the student model features, we draw insights about student behavior relevant to learning, and to course design.

8.1 Student Model Features

We group the course features from student interactions into four broad categories: 1) course behavior, 2) forum behavior, 3) coaching environment, and 4) topic analysis of forum posts. We extract features from student course behavior and forum behavior, which we describe in Sections 4 and 5. The two other feature categories are described below.

8.1.1 Coaching Environment

Students in the online course are either coached or independent. Coaches are provided a separate discussion forum, apart from the student forum, where they can interact with other coaches and instructors of the course. We extract features that capture coaches’ prior knowledge and their involvement in guiding students. Table 5 gives the list of coaching related features extracted from the discussion forum for coaches.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coached</td>
<td>Boolean feature capturing whether a student is coached or independent</td>
</tr>
<tr>
<td>Coach Views</td>
<td># posts viewed by the coach</td>
</tr>
<tr>
<td>Coach Questions</td>
<td># questions posted by the coach</td>
</tr>
<tr>
<td>Coach Answers</td>
<td># answers posted by the coach</td>
</tr>
<tr>
<td>Coach Contributions</td>
<td># contributions in the forum</td>
</tr>
</tbody>
</table>

Table 5: Coaching related features

8.1.2 Posts Topic Distribution

For extracting topics of the post, we explore the topic modeling framework using Latent Dirichlet Allocation (LDA) [3]. Before using LDA we clean the text data by removing stop words, stemming certain words, and removing all common course words, such as code. To obtain the topic distribution of posts, we use the Machine Learning for Language Toolkit (Mallet) [7]. We use the following parameters for the topic model: number of topics = 150, and optimize-interval = 100, where the hyper-parameters required by LDA, $\alpha$ and $\beta$, are set to the default values.

8.2 Predictive Model

We incorporate extracted features in a linear kernel Support Vector Machines (SVM) model, using the python package Scikit-learn [10]. Comparing this model with other machine learning algorithms such as logistic regression, decision trees, and Naive Bayes we found the results to be comparable. We filter our student pool to those who participated in the forums and took the post test (approximately 16% of all students who completed the post test). A subset of features that are predictive of post-test performance were selected using recursive feature elimination in Scikit-learn [10]. Recursive feature elimination works by training a classifier which weighs features and then trims all features with the lowest weights; this trimming allowed us to obtain the best predictions, and to understand which features are most predictive of student success.

8.3 Empirical Results

In this section, we present empirical results using the SVM model defined above to predict post-test performance. To evaluate the effectiveness of this model we compute the F-measure, which is the harmonic mean of precision and recall. F-measure is an optimal metric for a setting with unbalanced classes such as ours, where accuracy may appear to be deceptively high if a classifier reliably predicts the majority class. Our model gives an F-measure of 0.81 for predicting post-test performance. We validate our results with 10-fold cross validation. In the next sections, we analyze the attributes of student behavior which are most predictive of performance.

8.3.1 Topics and Performance

The topics discovered by the topic model fall into four broad categories: help requests, assignments, course material, and course activities. In Table 6, we present the top ten topics which are most predictive of post-test performance. The first three topics in the table fall into the help requests category. They include words such as trouble, help, and fail. Four of the top ten topics correspond to assignments, with top words which are descriptive of assignments from the course. For example, in assignment $A_1$ students are asked to write a program to count the number of hashtags, links, and attributions in a tweet, and in the topic associated with this assignment we see the words: hashtag, tweet, attributions, mentions, and links. Two topics represent the concepts discussed in the course: object oriented programming, and hash maps. The hash maps topic is particularly interesting as hash maps are not introduced in the course, but students still use them in their projects, and discuss them on the forum. The other prominent topics are topics related to course activities. For example, the activity topic in the the table is an activity given to students to print the location of a vehicle. This is the most elaborate activity that students undertake in the course, hence it appears in the top predictive topics for predicting post-test performance.

<table>
<thead>
<tr>
<th>Topic Label</th>
<th>Top Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help requests</td>
<td>trouble, don't, perfectly, won, updated</td>
</tr>
<tr>
<td>Help requests</td>
<td>help, helps, change, find</td>
</tr>
<tr>
<td>Assignment content ($A_1$)</td>
<td>hashtag, tweet, attributions, mentions, links</td>
</tr>
<tr>
<td>Lecture (hashmaps)</td>
<td>Map, key, iteration, Hashmap, entry</td>
</tr>
<tr>
<td>Course Activity</td>
<td>vehicle, location, backward, forward, getLocation</td>
</tr>
<tr>
<td>Assignment content ($A_2$)</td>
<td>Array, list, words, remove, equals, size</td>
</tr>
<tr>
<td>Assignment content ($A_3$)</td>
<td>set, insertion, swap, insert, algorithm</td>
</tr>
<tr>
<td>Lecture (OOP and Methods)</td>
<td>object, constructor, methods, parameter, return</td>
</tr>
</tbody>
</table>

Table 6: Top predictive topics and the words in these topics

Figure 8 gives the distribution of passed and failed students across the different ten most predictive topics given in Table 6. We observe that passing students post about the course activity on vehicles more than failing students. Since activities only contribute to a small portion of their grade,
participation in activities is a good measure for students’ level of motivation and learning.

Additionally, we observe that failing students are far more likely to write posts which fall in the help category. Looking at some of the posts in this category, we find that these posts are often short and use help words, but do not contain detailed information about the specific assignment problem in question. This finding suggests that analyzing the posts for linguistic cues is helpful in understanding students’ motivation.

The third important take away from this analysis is that this topic distribution can help discover patterns in student behavior. For example, passing students post about assignment A10 more than failing students. But, failing students post more about assignment A4. As assignments tend to get harder as the course progresses, the difference in behavior can be attributed to failing students needing help on the easier assignments, while the savvier students focus on the harder assignments.

8.3.2 Critical Assessments
Here, we describe the most predictive assignments, quizzes and exams that we use in the predictive model. We find that assignments A4, A8, A9, and A10 are the most predictive assignments. These assignments are on core concepts and hence may be the most critical assignments in the course. This observation is bolstered by the fact that these assignments are referenced in the forums more than other assignments. Two of these assignments feature in the top ten predictive topics given in Table 6. Pinpointing the moment when a student needs help is not only predictive of their success, but also critical in maintaining engagement and understanding. Understanding which assignments are discussed more in the forums can reveal important information for initiating instructor interventions.

9. CONCLUSION
From this analysis we conclude that MOOCs are a viable option for high school students. Forty-seven percent of students who took the post test passed it. Four hundred and sixty-four of these students were to the best of our knowledge self-directed. While we can say that MOOCs work for some high school students, the particularities of this group must be understood. It is not clear, for example, how the students who achieve high course scores, but low AP exam scores are able to do so. Are they receiving answers from other students, or have they truly mastered the course content, but lack the ability to demonstrate this mastery on a test? High school MOOC students are a unique group with particular modeling demands.

We have developed models of these students, characterizing high and low learners by their course and forum behavior, as well as by the topics that they post about. These models have allowed us to differentiate the behavior of students who pass from that of students who fail. In this case study post-test performance was correlated with course-performance, such that students who earned a high course score also earned a high post-test score. Students who performed well on the post test were more likely to contribute to conversations, and to answer questions on the student forum. They were also more likely to post about ungraded activities, and less likely to write posts asking for help. Coached students were more likely to perform well in the course, and spent more time on the forum. Understanding the differences between students who excel and those who do not is crucial in developing the courses that students, and particularly high school students need.

References
The Affective Impact of Tutor Questions: Predicting Frustration and Engagement

Alexandria K. Vail
Department of Computer Science
North Carolina State University
Raleigh, North Carolina
akvail@ncsu.edu

Joseph B. Wiggins
Department of Computer Science
North Carolina State University
Raleigh, North Carolina
jbwiggj3@ncsu.edu

Kristy Elizabeth Boyer
Department of Computer & Information Science & Engineering
University of Florida
Gainesville, Florida
keboyer@ufl.edu

Eric N. Wiebe
Department of STEM Education
North Carolina State University
Raleigh, North Carolina
wiebe@ncsu.edu

James C. Lester
Department of Computer Science
North Carolina State University
Raleigh, North Carolina
lester@ncsu.edu

ABSTRACT
Tutorial dialogue is a highly effective way to support student learning. It is widely recognized that tutor dialogue moves can significantly influence learning outcomes, but the ways in which tutor moves, student affective response, and outcomes are related remains an open question. This paper presents an analysis of student affective response, as evidenced by multimodal data streams, immediately following tutor questions. The findings suggest that students’ affect immediately following tutor questions is highly predictive of end-of-session self-reported engagement and frustration. Notably, facial action units which have been associated with emotional states such as embarrassment, disgust, and happiness appear to play important roles in students’ expressions of frustration and engagement during learning. This line of investigation will aid in the development of a deeper understanding of the relationships between tutorial dialogue and student affect during learning.

Keywords
Tutorial dialogue, affect, frustration, engagement, facial expression

1. INTRODUCTION
Tutorial dialogue provides rich, natural language adaptation to students during learning. An understanding has emerged about the role of interactivity in tutorial dialogue [40, 6] and on dialogue strategies for most effectively supporting students in task-oriented tutorial dialogues [29, 10]. However, a pressing issue is developing an understanding of how specific tutor dialogue moves impact students’ affect, and in turn, what influence students’ affective responses may have on outcomes.

The need for modeling affect during learning is widely recognized. Research has shown that suites of affect detectors from sensors and log files can perform well but that there are trade-offs depending on the goals of the affect detection modules [22, 33]. Affect detectors have been investigated for a wide variety of affective states including confidence, excitement, frustration, and interest [41], and within tutorial dialogue, for uncertainty [11]. There have also been great strides in sensor-free affect detection which relies primarily on log files [2]. This approach has shown promise during cognitive tutoring [9] and for distinguishing frustration and confusion [27].

Out of all of the affective phenomena that have been examined during learning, two affective states are frustration and engagement. These states have been examined in fine-grained analyses as tutoring unfolds, and also as outcome measures regarding students’ perceptions of the success of the tutoring session. Engagement and frustration have been predicted at above-chance levels using facial expression-based affect detection even without the presence of interactive events during text or diagram comprehension [5]. Engagement and frustration have also been predicted with nonverbal behaviors, including facial expression, after student task events during problem solving [16]. In a compelling development, emerging evidence shows that fine-grained affective events can have long-lasting relationships with outcomes that may be far removed from those affective events [36].

This paper advances the understanding of student emotions in learning by examining students’ fine-grained affective responses to tutor questions during tutorial dialogue. It investigates the hypothesis that students’ affective responses immediately following tutor questions are related to self-reported frustration and engagement at the end of the session. The results indicate that several key facial expression...
features immediately following two different types of tutor questions are highly predictive of end-of-session self-reported engagement and frustration. This line of investigation represents a step forward in understanding the affective impact of tutorial strategies.

2. RELATED WORK

Tutorial dialogue researchers have long studied what human tutors naturally do: how strategies differ between experts and novice tutors [12] whether Socratic or didactic approaches are most effective [35] and how tutors scaffold and fade support during problem solving [4], among others. The impact of particular tutorial dialogue moves has been the focus of significant attention, with findings indicating that positive and negative feedback have different impact based on students’ self-efficacy level [3], that bottom-out directives are not conducive to learning [29], and that adapting to student uncertainty improves the effectiveness of tutorial dialogue [10]. However, this paper examines a different aspect of these tutorial dialogue moves that is critical in learning: students’ affective response as expressed on the face and as embodied in gestures.

Multimodal features such as dialogue, facial expression, posture, and task actions have been used to predict affective states, such as boredom, confusion, excitement, and frustration, as those states occur during learning [23, 8, 7]. Moreover, multimodal features such as facial expression and gestures can significantly predict frustration and engagement reported at the end of tutoring sessions [17], and some differences have emerged in the extent to which upper and lower facial expression features are associated with these outcomes [15]. This previous work on utilizing multimodal features for predicting frustration and engagement during human-human tutoring has emphasized the important role that tutor dialogue moves play in affective outcomes. Other factors, such as student personality profile, can also contribute significantly to predicting these outcomes [39]. The present work examines moment-by-moment affect as evidenced by multimodal traces, and then analyzes the relationship between these multimodal behaviors and the outcomes of frustration and engagement as reported by students after the tutoring session.

3. STUDY DATA

The present analysis investigates the multimodal behavior of students during a computer-mediated tutorial session in introductory computer science, and specifically in Java programming [18, 30]. The tutorial interface, shown in Figure 1, is divided into four panes: the task description, the student’s Java source code, the compilation and execution output of the program, and the textual dialogue messages between the tutor and the student. The tutor’s interactions with the environment were constrained to progress between tasks and sending textual messages to the student.

Students ($N = 67$) were university students in the United States enrolled in an introductory engineering course, with an average age of 18.5 years ($s = 1.5$ years), whereas the human tutors ($N = 5$) were primarily graduate students with previous experience in tutoring or teaching introductory programming. The behavior of the student was collected using a set of multimodal sensors, as shown in Figure 2, including a Kinect depth sensor, an integrated webcam, and a skin conductance bracelet. The following subsections detail the modalities appearing significant in the present analysis.

Each student participated in six 40-minute sessions over the course of four weeks; however, the present analysis only examines data from the first lesson. Before and after each lesson, students completed a content-based pretest and identical posttest; the tutoring sessions were found to be significantly effective in facilitating learning gains ($p < 0.0001$). In addition to the posttest, students also completed a post-survey, including the NASA-TLX workload survey [20] and the User Engagement Survey [32]. The present analysis investigates self-reported frustration, taken from the Frustration Level item of the NASA-TLX workload survey, and engagement, taken as an average of three sub-scales of the User Engagement Survey: Focused Attention (perception of time passing), Felt Involvement (perception of involvement with the session), and Endurability (perception of the activity as worthwhile).

3.1 Task Event and Dialogue Features

During the tutoring session, the interface described above logged tutor and student dialogue messages, student typing in the code window, and student progress through the task. No turn-taking measures were enforced in the dialogue: students and tutors could send messages to the other at any point. All exchanged messages were automatically tagged by a J48 decision tree classifier [37] with a dialogue act annotation scheme created for task-oriented tutorial dialogue that differentiates tutor questions, feedback, and hints, among other dialogue moves [38]. In that work, the Cohen’s kappa between two human annotators was 0.87 and the Cohen’s kappa between human and the J48 decision tree classifier was 0.786.

The analysis presented here focuses on two types of tutor dialogue moves: inference questions and evaluative questions. (Although other question types were investigated, student reactions to these were not found to have significant predictive power.) Inference questions require the formation of an action plan or reasoning about existing content knowledge. For example, ‘How do you think this problem can be solved?’, or ‘How can you fix this error?’ are considered to be inference questions. On the other hand, evaluative questions aim to evaluate the student’s belief in his or her own understanding of the material, e.g., ‘Does that make sense so far?’ or ‘Do you understand?’ (see Figure 4).

Previous work has suggested that questions can stimulate cognitive disequilibrium in a student [34], which is often considered to be a critical step in knowledge acquisition [13]. On the other hand, evaluative questions that ask a novice to evaluate whether she understands material may not be particularly helpful pedagogically because novices often cannot identify what they do not understand, or may be hesitant to speak up even if they are aware that they are confused. Nonetheless these questions occurred regularly in our corpus with experienced (though not expert) human tutors. We investigate whether students’ affective response to these types of tutor dialogue moves is significantly predictive of student engagement and frustration as reported at the end of the session.

3.2 Facial Expression Features

Student facial expressions were automatically extracted
using a state-of-the-art facial expression recognition toolbox, FACET (commercial software preceded by a research version known as the Computer Emotion Recognition Toolbox, CERT) [26]. FACET tracks the frame-by-frame presence of several facial action units according to the Facial Action Coding Scheme [25]. These action units include movements such as AU6 Cheek Raiser, AU12 Lip Corner Puller, AU24 Lip Pressor, and AU26 Jaw Drop (see Figures 5 and 6 for illustration). For each facial action unit, the FACET software suggests an Evidence measure, indicating the chance that the target expression is present. This Evidence measure is on a scale where negative values represent evidence of the absence of a facial expression and positive values indicate evidence of the presence of one. The more positive the measure, the more confident FACET is that the feature is present.

3.3 Gesture Features

The Kinect depth camera also tracked hand-to-face gestures made by the student during the tutoring session. An algorithm developed to detect such gestures was developed to recognize one or two hands touching the lower face. In order to do this, the algorithm relies on surface propagation from the center of the head, identifying round (i.e. a normal head shape) or oblong shapes (i.e., shapes extending beyond the normal head shape) based on distances from the center of the head. This gesture detection algorithm was previously found to be 92.6% accurate when compared against manual labels [14].

4. ANALYSIS

The present analysis focuses on the affective response of a student, as observed by multimodal traces of face and gesture, after tutor inference questions and evaluative questions. We hypothesize that multimodal features after these tutor questions can predict student engagement and frustration. In particular, we examine three seconds after each tutor dialogue move (a manually-determined interval). The multimodal response of the student was characterized using the following categories of features, all of which were provided to the predictive models. However, note that only the first two of these categories of features (shown in bold below) appear significantly predictive within the models.

1. Average evidence measure for each of the facial expression action units during the interval (19 features)
2. Percentage of the interval in which a one-hand-to-face or two-hands-to-face gesture was observed (2 features)
3. Number of skin conductance responses identified during the interval as measured by a skin conductance response bracelet (1 feature)
4. Average student distance from the workstation during the interval (1 feature)
5. Average difference between the highest and lowest points of the student’s body from the workstation during the interval, indicating leaning (1 feature)

We calculated the average value of each multimodal feature listed in the categories above across each tutoring session. For each feature, we computed its conditional probability of occurring after the tutor moves of inference question or evaluative question. We also provided the model with the overall occurrence of that feature across the entire tutoring session.
Figure 2: Multimodal instrumented tutoring session, including a Kinect depth camera to detect posture and gesture, a webcam to detect facial expression changes, and a skin conductance bracelet to detect electrodermal activity.

Figure 3: Dialogue excerpt illustrating a tutor inference question in context.

**Student** compiles the program, encounters an error.

<table>
<thead>
<tr>
<th>Student</th>
<th>Oh.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tutor</td>
<td>So how can we fix this?</td>
</tr>
<tr>
<td>Student</td>
<td>Hmm.</td>
</tr>
<tr>
<td>Student</td>
<td>Switch the prompt line with the response line?</td>
</tr>
<tr>
<td>Tutor</td>
<td>Okay, try it.</td>
</tr>
</tbody>
</table>

5. RESULTS AND DISCUSSION

For both types of tutor question, evaluative and inference, a predictive model was built to predict student frustration and student engagement, resulting in a potential four models. Three of the four models uncovered significant predictive relationships. The following subsections detail models predicting frustration after tutor inference and evaluative questions, and a model predicting engagement after tutor evaluative questions.

5.1 Frustration

The results suggest that student facial expressions are significantly predictive of self-reported end-of-session frustration. The predictive model for student frustration based on tutor evaluative questions includes two features, both of which are facial action units occurring in the three-second interval following the tutor evaluative question (Table 1).

Two facial action unit features after tutor evaluative questions in order to control for the influence of the feature overall (rather than only after the tutor moves of interest). Specifically, the features conditional on tutor moves were averages of the form $\text{Avg}(\text{Feature}|\text{TutorQ})$ for each student that completed the session. The session-wide average of each feature, $\text{Avg}(\text{Feature})$ were also provided to the model for each multimodal feature in all of the categories above.

Standardization was performed on each feature by subtracting the mean and dividing by the standard deviation, so that the regression coefficients would be more interpretable. The standardized features were provided to a stepwise regression modeling procedure optimizing for the leave-one-student-out cross-validated $R^2$ value (the coefficient of determination), while at the same time requiring a strict $p < 0.05$ cut-off value after Bonferroni correction on significance values.

$^1$The models reported in this paper were built as a part of a larger exploratory analysis. As a result, the p-values reported have been modified by a Bonferroni correction...
Table 1: Predictive model for standardized end-of-session frustration after tutor evaluative questions (TutorQE).\(^1\)

<table>
<thead>
<tr>
<th>Frustration =</th>
<th>R(^2)</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.7039 * AU12 after TutorQE</td>
<td>0.0764</td>
<td>0.014</td>
</tr>
<tr>
<td>-0.6279 * AU28 after TutorQE</td>
<td>0.2471</td>
<td>0.030</td>
</tr>
<tr>
<td>-0.1635 (Intercept)</td>
<td>1.000</td>
<td></td>
</tr>
</tbody>
</table>

Leave-One-Out Cross-Validated R\(^2\) = 0.3235

Table 2: Predictive model for standardized end-of-session frustration after tutor inference questions.\(^1\)

<table>
<thead>
<tr>
<th>Frustration =</th>
<th>R(^2)</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0.5660 * AU6 after TutorIQ</td>
<td>0.2893</td>
<td>0.022</td>
</tr>
<tr>
<td>+0.3635 * AU20</td>
<td>0.0499</td>
<td>0.019</td>
</tr>
<tr>
<td>-0.0174 (Intercept)</td>
<td>1.000</td>
<td></td>
</tr>
</tbody>
</table>

Leave-One-Out Cross-Validated R\(^2\) = 0.3392

Facial actions are significantly predictive of student frustration. Higher intensity levels of of AU12 Lip Corner Puller (Figure 5b) following a tutor evaluative question are negatively indicative of frustration, as is the presence of AU28 Lip Suck (Figure 5d). AU12 is associated with smiling, which is typically not associated with frustration although on occasion, the two can go hand in hand [21].

AU 28 is a type of lower face movement sometimes associated with fidgeting, and this type of motion may be a "self-manipulator" that is part of emotion regulation. It is possible that students engaged in this challenging learning task may exhibit this movement to alleviate negative emotions related to frustration, resulting in lower self-reported frustration at the end of the session. When students are faced with a question that asks them to evaluate whether they understand the material being tutored, these facial expressions may both reflect the presence of emotion regulation that could mitigate the students’ overall feeling of frustration.

The next model examines student responses to tutor inference questions. In contrast to evaluative questions, inference questions ask students to bring pieces of knowledge together to infer the answer to a question and then to express a substantive answer. Two facial action unit features exhibited following these questions appear as significantly predictive of student frustration. The model shows that AU6 Cheek Raiser (Figure 5a) after tutor inference questions is positively predictive of frustration, as is the overall session occurrence of AU20 Lip Stretcher (Figure 5c). The model is displayed in Table 2.

Interestingly, AU6 has been related to pain expressions in the literature on pain detection [28]. When asked to answer an inference question, it is possible that students exhibited a "pained" expression that coincides with frustration. The expression of AU20 has been observed to coincide with moments of embarrassment or awkwardness [24], when people were embarrassed or amused in the period after doing directed facial actions (the technique used to develop images for the Facial Action Coding System). AU20 only occurred among embarrassed participants in that study. When faced with a tutor inference question, this expression may indicate that the student is unsure, awkward, or embarrassed, which may unsurprisingly be related to frustration. Deeper future investigation of subsequent student dialogue moves will help elucidate this phenomenon.

5.2 Engagement

Next we built models to predict student engagement based on affective responses to tutor inference questions and evaluated\( p \leq \alpha/n \), where \( n = 21 \) is the number of statistical tests conducted in the larger analysis, in order to reduce the familywise error rate to \( \alpha = 0.05 \).
3. Predictive model for standardized engagement after tutor evaluative questions.1

<table>
<thead>
<tr>
<th>Engagement =</th>
<th>$R^2$</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0.4422 * ONEHTF</td>
<td>0.1815</td>
<td>&lt; 0.001</td>
</tr>
<tr>
<td>−0.5989 * AU10 after TUTOREQ</td>
<td>0.1831</td>
<td>&lt; 0.001</td>
</tr>
<tr>
<td>+0.5770 * AU12</td>
<td>0.2280</td>
<td>&lt; 0.001</td>
</tr>
<tr>
<td>+0.5097 * AU26 after TUTOREQ</td>
<td>0.0514</td>
<td>&lt; 0.001</td>
</tr>
<tr>
<td>−0.2941 * AU2</td>
<td>0.1923</td>
<td>0.003</td>
</tr>
<tr>
<td>+0.2467 * AU5</td>
<td>0.0295</td>
<td>0.002</td>
</tr>
<tr>
<td>+0.1792 * AU24 after TUTOREQ</td>
<td>0.0566</td>
<td>0.018</td>
</tr>
<tr>
<td>+0.4100 (Intercept)</td>
<td>1.000</td>
<td></td>
</tr>
</tbody>
</table>

Leave-One-Out Cross-Validated $R^2 = 0.9224$

The model is selected. The more frequently a student was displaying a **OneHandToFace** gesture, which may indicate thoughtful contemplation, the more engaged the student reported the experience at the end of the session.

Three more session-wide facial expression features were selected as significantly predictive of student engagement. The more intense the expression of AU12 **Lip Corner Puller** (Figure 5b) or AU5 **Upper Lid Raiser** (Figure 6b), the more engaged the student. For AU12 which is often associated with smiling, a positive emotion is likely related to higher engagement. In this task, AU5 is likely associated with the student looking at the screen, possibly indicating paying attention and focusing on the task (as opposed to the opposite facial movement of blinking or shutting one’s eyes). In contrast, AU2 **Outer Brow Raiser** (Figure 6a) was predictive of lower engagement. This action unit is a component of the “fear brow” (AU1+2+4) which has been evidenced as a display of anxiety [19].

Narrowing down to the context of three seconds after tutor evaluative questions, three facial expression features were significantly correlated with student engagement. The more that a student expresses AU26 **Jaw Drop** (Figure 6e), or the more that the student expresses AU24 **Lip Pressor** (Figure 6d), the more engaged the student reported being at the end of the session. Jaw drop is a dynamic action unit that may occur when the mouth is closed or already partly open. In either case, this action unit may be associated with focus on the task, although it could also plausibly be associated with a yawn (which we would not expect to coincide with higher engagement). With respect to AU24, which is a prototypical component of anger, an important interplay of learning and affect expression emerges. Some facial movements that are part of prototypical displays of negative basic emotions, such as anger, appear to be indicative of mental effort during learning, rather than negative affect [31]. From this perspective, it makes sense that this AU24 would be related to engagement. On the other hand, the more that a student expressed AU10 **Upper Lip Raiser** (Figure 6c) during this interval, the less engagement reported by the student at the end of the session. This action unit, which is a component of prototypical disgust, is likely to run contrary to engagement.

6. CONCLUSION

Tutor dialogue moves in one-on-one human tutoring significantly influence student outcomes, both cognitive and affective. This paper has examined students’ affective response to two types of tutor questions: inference questions which require some reasoning to construct an answer, and evaluative questions, which ask students to reflect on the extent to which they understand the material. The results show that immediately after these tutor questions, students’ affective displays—particularly with respect to facial expression—are highly predictive of the outcomes of frustration and engagement. By detecting these affective displays which have been associated in prior studies with emotions such as embarrassment, disgust, or happiness, we can begin to understand the moment-by-moment affective processes that influence learning through tutorial dialogue, and relate those fine-grained events to overall outcomes.

While these facial movements have been associated with prototypical emotion displays in the literature, it is important to further contextualize the moments in which these expressions appear during tutoring. For instance, action units typically associated with anger are likely indicators of mental effort during learning. Similarly, an action unit associated with disgust (e.g., AU10) may be related to students’ appraisal of the tutor’s question in the moment. Further research seeks to ground these interpretations more extensively across salient moments of tutoring.

There are several additional directions for future work. Detecting important moments during tutoring is an open area of investigation, with evidence suggesting that moment-by-moment affect may be related to distal outcomes [36, 1]. In future work, it will be important to expand our understanding of the identified non-verbal predictors for frustration and engagement more deeply. We must consider a wider variety of contexts, and explore different widths of time after tutorial events to examine affective responses with longer (or shorter) times to manifest. It is hoped that this line of investigation will lead to richer affect models for tutorial dialogue.

Acknowledgments

The authors wish to thank the members of the LearnDialogue and Intellimedia groups at North Carolina State University for their helpful input. This work is supported in part by the Department of Computer Science at North Carolina State University and the National Science Foundation through Grants IIS-1409639, CNS-1453520, and a Graduate Research Fellowship. Any opinions, findings, conclusions, or recommendations expressed in this report are those of the participants, and do not necessarily represent the official views, opinions, or policy of the National Science Foundation.

7. REFERENCES


C. M. Mitchell, E. Y. Ha, K. E. Boyer, and J. C. Lester. Learner characteristics and dialogue: recognising effective and student-adaptive tutorial


Figure 6: Sample frames from the student webcam illustrating the facial action unit features appearing in the predictive model for student engagement, as identified by FACET. Note that AU12 Lip Corner Puller (Figure 5b) also appears in these models.
ABSTRACT
Graph data such as argument diagrams has become increasingly common in EDM. Augmented Graph Grammars are a robust rule formalism for graphs. Prior research has shown that hand-authored graph grammars can be used to automatically grade student-produced argument diagrams. But hand-authored rules can be time consuming and expensive to produce, and they may not generalize well to novel contexts. We applied Evolutionary Computation to automatically induce empirically-valid graph grammars for argument diagrams that can be used for automatic grading or provide the basis for hints. Our results show that our approach can generate more relevant rules than experts or other state of the art algorithms, and that these evolved rules outperform the alternatives.

Keywords
Evolutionary Computation, Augmented Graph Grammars, Argument Diagramming, Feature Engineering

1. INTRODUCTION
Intelligent tutoring systems and computer-supported collaboration platforms have grown increasingly popular in recent years. As they have grown in popularity they have also been applied in increasingly complex domains such as argumentation [14], legal reasoning [22] and writing [6]. MOOCs and other online educational platforms have also grown in popularity yielding large repositories of user-system interaction logs [10], and classical tutors and educational games have grown more common in classrooms yielding large repositories of student data [13]. Much of this data can be represented as rich graph structures such as argument diagrams [17] or interaction networks [7].

Despite the increasing prevalence of graph data, comparatively little work has been done on automatically evaluating student-produced graphs or graph logs. In prior work we demonstrated that hand-authored Graph Grammars can be used as features to automatically grade student-produced argument diagrams [16, 17]. But hand-authoring complex rules is time consuming, expensive, and does not generalize well to novel contexts. Other authors have developed analytical tools tuned to path analysis [24, 3], however these are tailored to a specific task. Other more general purpose algorithms (e.g. [30, 5]) have limitations and are unsuited to the induction of generalized rules that use negation or other complex elements. Therefore it has not yet been shown that it is possible to automatically induce complex, empirically-valid, rules for rich graph structures that are comparable to rules produced by domain experts.

In this paper we will describe our work on the automatic induction of Augmented Graph Grammars for student-produced argument diagrams. Our goal in this work is to explore ways to automatically induce empirically-valid graph rules that can be used as features for automatic grading and which can provide the basis for hints. While our previous work was focused on inducing positive rules in [33] and in [19], in this work we applied Evolutionary Computation (EC) to induce both positive and negative rules for student graphs that incorporate more complex elements such as negation and generalized types. Additionally, in our previous work we compared the induced rules with a small number of expert rules while in this work, we will compare our induced rules to a full set of complex rules authored by domain experts and rules produced by other state of the art induction algorithms.

2. BACKGROUND
2.1 Argument Diagrams
Argument diagrams are semi-formal graphical representations that reify key features of arguments such as hypothesis statements, claims, and citations as nodes and the supporting, opposing, and clarification relationships between them as arcs. Argument diagrams directly connect the syntax of the argument representation to the underlying semantics thus making it clear and computationally tractable. Argument diagrams can serve to make the often implicit structure of an argument salient to students while also constraining them to make relevant contributions [29]. Prior researchers have shown that argument diagrams can be used to scaffold students' understanding of existing arguments [12, 8]; can frame collaborative learning [26]; and can help to support scientific reasoning [29].
A sample student-produced diagram is shown in Figure 1. The diagram includes a central research claim node, which has a single text field indicating the content of the research claim. A set of citation nodes are connected to the claim node via supporting, opposing and undefined arcs colored green, red, and blue respectively. Each citation contains two fields: one for the citation information, and the other for a summary of the work; each arc has a single text field explaining what purpose the relationship serves. At the bottom of the diagram, there is a single isolated hypothesis node that contains two text fields, one for a conditional or IF field, and the other for a consequence THEN field.

2.2 Augmented Graph Grammars

Graph Grammars are a graph-based representation for rules about graphs that are analogous to string grammars. Graph grammar rules are composed of standard graph elements such as nodes and directed or undirected arcs. As with string grammars they are defined by a finite alphabet of basic or ground node and arc types as well as a set of production rules for variable elements. A single graph rule defines a space or class of matching graphs. Graph grammars can be used to generate graphs from an initial seed via recursive rule applications where each variable element expands to a larger subgraph. They can also be used to match graphs in a layered fashion by first mapping all ground elements to individual nodes or arcs and then recursively matching the sub-elements. Graph grammars have been used for analysis and graph transformation in domains such as visual programming [9] and mechanism analysis [27].

Augmented Graph Grammars are an extension of traditional graph grammars that allow us to match rich graphs with complex node and arc types that contain sub-elements, text, and other variable structures [15]. Augmented Graph Grammars also support: negated elements which select for the nonexistence of subgraphs; generalized node and arc types which match multiple items; complex element constraints which allow us to compare individual elements; complex graph expressions which allow for universal and existential quantification; and the incorporation of NLP rules or other external features. As such they are an ideal rule representation for the analysis of argument diagrams, user-system interaction logs, and other educational data.

A sample rule is shown in Figure 2. This rule is designed to identify cases of uncompared counterarguments, that is: there is an opposing arc $O$ from the citation $a$ to the node $t$ and also a supporting arc $S$ from the citation $b$ to the node $t$, however, there exists no comparison arc between the two citations $a$ and $b$. This is designated by the negated arc $\neg c$. Here node $t$ is either a claim or hypothesis. The variable elements $O$ and $S$ are defined by recursive production rules which are not shown. Those rules define supporting paths as chains of supporting arcs and opposing paths as chains of supporting arcs with any odd numbered (including single) chain of opposing arcs.

\[
\text{(ParedWcomp)} \quad O \quad S
\]

\[
\begin{align*}
\{ & t.\text{Type} = \text{"claim"} \text{ or } \text{"hypothesis"} \\
& a.\text{Type} = \text{"citation"} \\
& b.\text{Type} = \text{"citation"} \\
& c.\text{Type} = \text{"comparison"} 
\}
\end{align*}
\]
This example rule was designed by a domain expert in argumentation. It is designed to identify cases where a student has presented conflicting background information but has made no attempt at resolution. This is a critical structural flaw that is commonly found in student-produced arguments. Students at all levels frequently absorb the lesson that they must show conflicting citations but routinely fail to explain those citations or to resolve the differences in a way that clarifies their own argument. As we have shown previously such expert-designed rules can be empirically-valid and predictive of student performance [16]. However manually designing rules can be both costly and inefficient.

Thus our goal is to automatically induce meaningful rules, rules that highlight structural flaws or argumentation errors; rules that generalize beyond basic types; and rules that include negated elements (detecting non-existing cases).

2.3 Graph Grammar Induction

Current grammar induction algorithms fall into one of two broad categories: frequent subgraph matching, or graph compression. Frequent subgraph algorithms include Yan and Han’s gSpan algorithm [32], Inokuchi’s AGM [1], and the FSG algorithm [20]. These algorithms carry out controlled graph walks to identify common structures. They are quite effective, particularly in grounded domains such as cheminformatics where the graphs, in this case molecular models, have low degree and exact matches are required. However the algorithms do not support disjoint subgraphs, negation, or generalized elements. While we can, in theory, insert explicit negation arcs that would expand the size of the graphs exponentially and thus make any search process intractable. Similarly, while we could replace individual elements with generalized forms that would simply force the system to use a smaller range of types and would not allow for context-sensitive generalization of elements. These algorithms are also ill-suited for identifying errors as the search process is strictly unsupervised and finds frequently-occurring structures without reference to external weights.

Graph compression algorithms such as Subdue take a different approach to the problem. Subdue is a recursive beam-search algorithm that generates a hierarchical grammar by recursive collapse based upon the MDL principle [5]. Subdue operates by iteratively identifying the most frequently occurring arc in the graph and then reducing it to a new variable node. Unlike gSpan the resulting grammar is hierarchical and the beam search process can be used for supervised learning given a suitable set of positive and negative examples [11]. The candidate graphs are ranked according to a normalized error metric:

\[
\frac{\text{PosGraphsNotCovered} + \text{NegGraphsCovered}}{\text{TotalExamples}}
\]

While Subdue is more flexible than the frequentist approaches it too does not support generalized elements, negation, or disjoint subgraphs.

2.4 Related Work

We have previously shown that domain experts can hand author augmented graph grammars that are empirically-valid and which can be used as features in a regression model to automatically grade student-produced diagrams [16, 17]. In more recent experiments we have also shown that it was possible to apply EC to induce graph grammars that are positively correlated with argument grades and that we can apply χ²-filtering to select unique rules from the large space of candidates [19]. We were also able to show that the induced rules outperformed rules generated by both Subdue and gSpan and outperformed similar expert rules that fit into the limited rule space. The rules produced in that study, however, were limited in scope. While they supported disjoint graphs, they did not identify errors, and did not support generalized elements or negation. In this work we will build upon these results to include generalization and negation, and we will compare the resulting rules to a full set of 77 hand-coded expert rules.

3. METHODS

We conducted two experiments on the induction of Augmented Graph Grammars using EC. First we applied EC to induce graph rules composed of static node and arc types that were both positively and negatively correlated with the overall argument quality. That is, we sought to identify ground rules that either highlighted good features of arguments (positive) or matched structural flaws (negative). We then compared them to expert-produced rules and to rules induced by the Subdue and gSpan algorithms. In our second experiment we applied EC to induce rules that also incorporated generalized nodes as well as negated arcs (detecting non-existing cases). We describe them below.

Evolutionary Computation is a general beam-search algorithm based upon Natural Selection. The EC algorithm begins with a population of candidate solutions in a shared solution representation. This population may be randomly generated or supplied by the user. The individual solutions are then ranked by means of a fitness function which may be an absolute performance metric or a form of tournament selection. The next generation of the population is then formed by a combination of fitness proportional selection, crossover or recombination of candidate solutions, random mutation of solutions, and elitist cloning. EC algorithms proceed iteratively until a given fitness threshold is reached or a fixed number of generations has passed. EC has been used in a number of applications such as tuning Neural Networks [21], and evolving computer code [2].

EC has a number of advantages over other special-purpose induction algorithms. Firstly, it is very flexible, the behavior of the system is determined by the user-specified solution representation and the genetic operators. This makes it easy to tune the behavior of the system to include new types of elements or to test out alternative inductive biases. Secondly, EC is very robust, the basic algorithm can be applied in a wide range of domains and it can be used in areas where the contours of the search space is unknown. There are a number of widely-available EC systems. For the purposes of this research we used pyEC an open-source EC engine [18] coupled with AGG an engine for graph matching using Augmented Graph Grammars [15].

The rules induced in Experiment I consisted entirely of ground nodes and arcs while the rules induced in Experiment II included generalized node types and negated comparisons as shown in Figure 2. For both experiments we assessed the
fitness of the rules using the same nonparametric frequency correlation that we discussed in Subsection 2.4 with the target values being maximized or minimized depending upon the experimental goals.

Mutation in the EC algorithm is a general-purpose operation that is designed to promote exploration by introducing heterogeneity into the population. For this set of experiments we applied basic point mutation that added, deleted, or modified individual graph elements (see [33, 19]). Here mutation occurred with a small constant frequency when individuals were added to each population.

For these experiments we employed stable matrix crossover based upon the work of Stone, Pillmore, & Cyre [28] illustrated in Figure 3. In this form of crossover we select a pair of parent graphs using fitness-proportional selection and represent them as adjacency matrices \((P_0)\). The nodes are represented by letters on the rows and columns, while the arcs are represented by the numbered cells within the table. Empty cells indicate the absence of an arc. The order of elements in the matrices is canonical and is determined by the order in which the nodes were added to the rule.

On crossover we align the nodes and arcs in the parent matrices and then randomly shuffle the nodes and arcs between them based upon a series of coin tosses to produce the two children \((C_0)\). Any constraints that are attached to an individual element are copied with it. Matrix crossover always produces two children that match the size of their parents with all excess elements being copied directly to the larger of the two offspring. Table 1 shows this crossover process at the graph level. By design crossover is an adaptive process that is designed to promote homogeneity and to preserve good building blocks or partial solutions called introns [2].

4. DATA

Our experimental analysis was based upon two previously-collected datasets. The first is a set of student-produced argument diagrams for empirical research reports. The second is a repository of hand-authored rules defined by domain experts. Both datasets were collected as part of our prior work on the diagnosticity of argument diagrams [16, 17].

4.1 Argument Data

Our repository of argument diagrams was collected at the University of Pittsburgh in a course on Psychological Research Methods. Students in the course learn about designing, conducting, and reporting on empirical research. The course has a significant writing component. Students complete two research projects over the course of the semester both of which result in a written report modeled on a conference publication. They are allowed to work on the projects individually or as a team of two. For the purposes of our study, the students were required to plan their written arguments graphically before writing them. The diagrams were authored using LASAD, an online tool for argument diagramming and collaboration [14]. The diagramming ontology contained four types of nodes: citation, claim, current study and hypothesis; and four types of arcs: supporting, opposing, comparison, and undefined. Current study nodes are used to represent factual information about the study such as the target population. Undefined arcs represent cases where nodes provide clarification or concept definitions.

After removing dropouts and one diagram containing a single node, we collected a set of 104 paired diagrams and essays from the course. These diagrams and essays were independently graded by an experienced TA according to a parallel rubric with 14 questions that were focused on the argument’s quality, coherence, use of citations, and other criteria. In this work we will focus on the gestalt grades for overall graph and essay quality. The gestalt grades were assigned on an 11 point scale from -5 (worst quality) to +5 (complete, coherent, and persuasive) at \(\frac{1}{2}\) point intervals. This same dataset was used in our prior work [19].

4.2 Expert rules

In parallel with data collection, we also collaborated with a group of domain experts to define a set of 77 a-priori argument rules. These rules were designed to identify individual features of argument diagrams or sub-graphs that were consistent with high quality argumentation or which represented structural flaws. Thirty-four of these rules focused on basic features such as the size or order of the diagram, the average number of parents and children, or the presence of empty elements. The remainder were complex rules that described the relationship between elements or matched larger graph structures such as the unmatched counterarguments shown in Figure 2. These rules included features that dealt with the text inside the elements, appropriate grounding of hypotheses or claims in citations, connectedness of the diagram, and the appropriate use of individual elements.

Table 1: Graphical representation for crossover.
In prior work we evaluated whether or not these rules were empirically-valid. That is whether or not they correlated with the independently-assigned diagram grades and whether or not they could be used to predict the paired essay grades [16, 17]. In that work we assessed the validity of each individual rule by testing the correlation between the observed rule frequency on each diagram and the final graph or essay grade. The strength of this correlation was assessed using Spearman’s ρ, a nonparametric correlation measure [31]. We found that most, but not all of the rules were strongly correlated with the grades. We also found that some of the correlations ran counter to the experts’ a-priori expectations.

5. EXPERIMENTS

In this work we induced sets of baseline rules using the Subdue and gSpan algorithms. We also conducted two sets of evolutionary experiments designated EC-Base and EC-General. The rules from each of these experiments were compared to assess their overall performance.

Subdue: For these experiments we used Subdue V5 [4] in supervised learning mode to induce rules that were positively and negatively correlated with the overall graph and essay grades. In order to induce positively correlated rules we partitioned the graphs into positive and negative examples based upon their graph or paired essay score. All graphs with a grade of 0 or more were treated as positive examples, and all graphs with a negative grade were treated as negative examples. We then ran the system to extract the 12 best rules. In order to induce negatively-correlated rules we reversed the assignment with rules that were graded less than or equal to 0 being treated as positive examples and all others being treated as negative. We experimented with more restrictive thresholds > 0 and < 0 and found the performance did not improve.

gSpan: In this experiment we used gSpan v6 [34]. The software runs in strictly unsupervised mode where it returns all subgraphs whose frequency exceeds a user-specified threshold. In this case we ran the software over our dataset and collected all rules that exceeded a 1% threshold and then ranked the candidate rules based upon their ρ value to identify the most positive and negative examples.

EC-Base: In this experiment, we conducted a series of six evolutionary runs that were tuned to induce negatively-correlated rules. Three of those runs used the graph grade as a target and three used the essay grade. In each case we used a fixed population size of 100 individuals and ran the algorithm for 1,000 generations. In each generation, we cloned the top 10 individuals directly into the next generation under elitism. We selected 10 individuals for point mutation and the remaining 80 individuals for crossover, then we copied the results over to the next generation. Fitness values were assigned using a fixed measure of −ρ for each individual rule. The initial populations were composed of randomly-generated individuals containing 3 - 10 elements each. The nodes and arcs were all ground elements and were selected from a predefined ontology of basic types that matched the types used in the argument diagrams.

Unlike standard EC we did not rely solely on the final population of rules for our results. EC populations grow increasingly homogeneous over time making the final population virtual clones. In this case our goal was to induce a range of potential rules. We therefore collected candidate rules from each generation of the run by selecting every rule with a ρ ≤ −0.1. The full set was used in our analysis.

EC-General: Here we conducted a series of twelve evolutionary runs. Six of the experiments were tailored to induce positively correlated rules while the rest were tailored to induce negatively-correlated ones. As with EC-Base the population size was 100, the algorithm ran for 1,000 generations, and we used ±ρ as the basic fitness metric and the mutation and crossover rate were the same as before. Unlike the EC-Base study these rules also included negated comparison arcs as well as two generalized node types: nodes that are citations or claims (CitOrClaim) and nodes that are hypotheses or claims (HypOrClaim). These elements were chosen for addition because they were used by the domain experts when crafting their rules. As before we collected candidate rules from the positive and negative runs with thresholds of (ρ ≥ 0.18) and (ρ ≤ −0.1) respectively. These thresholds were chosen based upon a series of exploratory runs in which we found that the ρ values became statistically significant after exceeding ±0.18.

6. RESULTS & ANALYSIS

Table 2 shows the number of positively and negatively correlated rules for the Graph grades (columns 3 and 4) and the Essay grades (columns 5 and 6) that were collected during our experiments. Total designates the total number of rules produced by each method or in the expert set, while Threshold indicates the number for which ρ ≥ 0.18 or ρ ≤ −0.18 in the positive and negative cases respectively.

As Table 2 shows the EC approaches generated the largest number of candidate rules in both the positive and negative cases. Of the expert rules, most of them were positively correlated with performance but less than half of them exceeded the cutoff thresholds. Indeed only two of the expert rules did so for the essay grades. Both Subdue and gSpan identified positively and negatively-correlated rules but only a few of the positive rules exceeded the threshold. None of the negative rules did so.

Next, we will describe the rules induced during our EC-Base

<table>
<thead>
<tr>
<th>Table 2: Number of Positive and Negative Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Methods</strong></td>
</tr>
<tr>
<td><strong>Methods</strong></td>
</tr>
<tr>
<td>Subdue</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>gSpan</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Expert</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>EC-B</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>EC-G</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

* Threshold: number of rules with ρ ≥ 0.18 or ρ ≤ −0.18
Table 3: Spearman correlation values for the best 3 rules in each experiment.

<table>
<thead>
<tr>
<th></th>
<th>Positive-correlated</th>
<th></th>
<th>Negative-correlated</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Graph 1st 2nd 3rd</td>
<td>Essay 1st</td>
<td></td>
<td>Essay 1st</td>
</tr>
<tr>
<td>Subdue</td>
<td>.276 .270 .253</td>
<td>.281 .215</td>
<td>.181</td>
<td></td>
</tr>
<tr>
<td>gSpan</td>
<td>.352 .314 .272</td>
<td>.300 .281</td>
<td>.261</td>
<td></td>
</tr>
<tr>
<td>Expert</td>
<td>.427*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC-B</td>
<td>.371 .369 .362</td>
<td>.334 .334</td>
<td>.319</td>
<td></td>
</tr>
<tr>
<td>EC-G</td>
<td>.396 .391* .385*</td>
<td>.357* .357*</td>
<td>.356*</td>
<td></td>
</tr>
</tbody>
</table>

* The best of results for Experiment I is in bold; *' is for best of results across both Experiment I and II.

experiment and we will discuss how they compare to the expert rules and the rules induced by Subdue and gSpan. We will then discuss the EC-General rules and compare them to our earlier results.

6.1 Experiment I: EC-Base

Rows 1-4 in Table 3 list \( \rho \) values for the three best rules from the four methods. The bold values indicate the best performing rule among the sets. As the table illustrates EC-B outperformed both Subdue and gSpan across the board. And it outperformed the expert rules in most cases. The lone exception being the best positive case for the graph grades and the best negative case for the essay grades.

The best positively-correlated expert rule for the graph grades matched arcs with empty text fields. The best negatively-correlated expert rule with the essay grade matched graphs with no hypothesis nodes. Both of these rules relied on complex grammar features, textual rules and expressions, that were outside the scope of our current experiments.

![Figure 4: EC-Base: Strongest Positively-correlated Rules Induced by EC.](image)

![Figure 5: EC-Base: Strongest Negatively-correlated Rules Induced by EC.](image)

Figures 4 and 5 illustrate the best positive and negative rules induced by the EC-Base runs. In Figure 4 graph rule B-G-P represents a rule that has 5-nodes, two of which are citations (c0 & c1) that support a shared claim node (k0). The remaining nodes are a single claim (k1) and a hypothesis (h) which may or may not be connected to the rest of the structure. This reflects a graph where the authors identified at least two related citations that can be synthesized to support a single claim and where they included both a hypothesis and another claim. This is one of the structures that students have been encouraged to make in their arguments as it shows an ability to synthesize citations to form a complex claim.

Interestingly, the best positive essay rule (B-E-P) is very closely related to the expert rule shown in Figure 2. Here it selects for the presence of a hypothesis node (h) that is directly connected to two citations (c0 & c1). Here c0 directly supports h while c1 directly opposes it. Given that the algorithm could not induce variable arcs it is not surprising that it does not include paths. The absence of a comparison arc, however, is interesting. As we noted above the students were instructed to include one. The fact that this rule performs so well despite lacking one suggests that the students did not regularly do so.

Figure 5 shows the best negative rules. As stated above, we expect that these rules will flag errors or persistent structural flaws. B-G-N consists of 4 claim nodes (k0 – k3) and two currstudy nodes (cS0 & cS1) all of which may or may not be connected to one-another. While this rule has a high correlation with the grade, its semantic meaning is unclear. It is possible that it is detecting is overly large graphs that lack sufficient focus. In future work we will evaluate the matching graphs with domain experts to assess this.

B-E-N is easier to interpret. In this case the rule contains a single claim node (k) which is connected to a citation node (c) via an undefined arc (u). This is a clear violation of the semantic guidance that students were given. The students in the experiment were instructed to use unspecified arcs for definitions or clarifications only. Some students instead
used them when they were unsure about the strength of their evidence or did not understand the citation. The students were also instructed to use citations to add information to their claims, not the other way around. For a student to use an unspecified arc in this way suggests that they were unsure about the structure or content of the argument.

6.2 Experiment II: EC-General

The last row of Table 3 shows the performance of the EC-General rules. These rules were compared against all of the rules in Experiment 1. The best performing rules across both experiments are in bold and marked *.

As Table 3 shows EC-General produced better performing rules than EC-Base. All but one of the \( \rho \) values on the final row exceeds the corresponding value on the fourth, and the one that does not do so falls behind by only 0.001. EC-General outperformed the best negative expert rule for the essay grades (-0.269 vs. -0.256), despite the fact that the expert rule relied on complex expressions. The best expert rule for the graph grade still outperforms EC-General. Thus, our results for EC are better than all other methods save for one expert rule that relies on novel textual features.

Figure 6 shows the best positively-correlated rules for the graph and essay grades. G-G-P matches cases where a supporting arc has been drawn from a citation or claim to a claim or hypothesis. In short, it matches correct uses of supporting arcs. This is a good feature that indicates well-supported arguments. G-E-P, by contrast, is complex and selects for a graph with three claim nodes (k0 – k2) and two uncompared citations (c0 & c1), where c1 directly supports a hypothesis or claim (hk) which in turn has an unspecified arc to a citation or claim node (ck). The semantic meaning of this rule is unclear and will require deeper analysis.

Figure 7 shows the strongest negatively-correlated rules. As with G-E-P, G-G-N, is somewhat hard to interpret. It selects for a number of disjoint nodes, and for the presence of a currstudy node (cs0) as well as a claim (k3) which are not connected via a comparison arc. Further analysis is required to determine why this rule holds. G-E-N, by contrast represents a clear variation on B-E-N. Here we select for a hypothesis or claim node (hk) that has an undefined arc to a citation along with a separate hypothesis node that may or may not be connected. This rule is interesting because in part it will select a superset of the graphs matched by B-E-N but the presence of the extra hypothesis node will restrict that somewhat. This suggests that this rule may be relatively specific to our dataset. We plan to examine the matching graphs to assess its generality.

7. CONCLUSIONS

In this paper, we reported our work on the automatic induction of Augmented Graph Grammars for student-produced argument diagrams through EC. In prior work we demonstrated that hand-authored expert rules can be empirically valid and that those valid rules can be used for automatic grading. We have now shown that it is possible to automatically induce complex rules for argument diagrams that match both positive and negative examples and which can therefore be used as features for automatic grading. We have also shown that the induced rules outperform all but one of the expert rules and the rules induced by other general-purpose grammar induction algorithms. The strongest expert rule was outside the scope of this experiment.

In future work we plan to work with domain experts to evaluate these rules. Our goal will be to determine whether the rules are semantically valid, and whether or not they can serve as the basis for automatic hints. We will also assess whether or not the rules can be used for data-driven grading by using them as features in a regression model. And finally we will expand the scope of our EC induction to include the automatic induction of hierarchical rules with expressions and complex element constraints.

8. REFERENCES


