Data-Driven Curriculum Design: Mining the Web to Make Better Teaching Decisions

Antonio Moretti†, José P. González-Brenes⋆, Katherine McKnight†
†Center for Educator Learning & Effectiveness
⋆Center for Digital Data, Analytics & Adaptive Learning
Research & Innovation Network, Pearson
{antonio.moretti, kathy.mcknight, jose.gonzalez-brenes}@pearson.com

ABSTRACT

University professors of conventional offline classes are often experts in their research fields, but have little training on educational sciences. Current educational data mining techniques offer little support to them. In this paper we propose a novel algorithm, Analyzing CurrIculum Decisions (ACID), that leverages collective intelligence to model student opinions in order to help instructors of traditional classes. ACID mines publicly available educational websites, such as student ratings of professors and course information, and learns student opinions within a statistical framework. We demonstrate ACID to discover patterns in learner feedback and factors that affect Computer Science instruction. We investigate the choice of a programming language for introductory courses and the grading criteria for all courses.

Keywords

offline teacher support, web mining, collective intelligence

1. INTRODUCTION

University professors of conventional offline classes are often experts in their research fields, but have little training on educational sciences. For example, studies have identified a lack of pedagogical training preparing research-based graduate students to teach in higher education [3]. It is not clear how existing educational data mining technologies can utilize the power of internet to learn student opinions in order to support traditional offline instructors.

We propose a novel algorithm, *Analyzing CurrIculum Decisions*(ACID), which is able to discover the effect of teaching decisions in the classroom by mining the increasing amount of information available online from educational websites. ACID develops resources and scripts to make use of collective intelligence and leverages this hierarchy of information within a statistical framework. ACID supports instructors of traditional offline courses by extracting from the web teaching syllabi data, and using crowd-sourcing to pair it up with students' course ratings and opinions to analyze the relationship between the two.

Algorithm 1 ACID pseucode

n universities to analyze, z reviews to analyze

procedure ACID

  while $|R| < z$ **do**

    $s \leftarrow$ sample of n universities

    $s \leftarrow$ Remove non-English speaking universities

    $R \leftarrow$ Search_The_Web_For_Reviews(s)

    $R \leftarrow$ ratings rated by more than ϵ students

  $Q \leftarrow$ CrowdSource_Questionnaire(R)

  Analyze_Data(Q)

This paper reports a case study of using the ACID methodology to answer questions that instructors of computer science courses face when designing their courses:

1. **For introductory classes, which programming language do students associate with clearer instruction?** The choice of a first programming language likely affects students' decision to continue education within the field of computer science. It is thus valuable to model data capturing learner sentiment.
2. **What grading rubric do students associate with clearer instruction?** Instructors want to optimize their grading criteria with respect to student learning and the student experience. The question of how to implement a grading rubric determines what students focus on within a course.

2. ANALYZING CURRICULUM DECISIONS

We use publicly available self-selected ratings of professors from a third-party website, *Rate My Professor* [4]. This site allows students to rate the professors and the courses they have taken. The website contains data from over 13 million ratings for 1.5 million professors. They collect ratings on a 1—5 scale (being 1 the lowest possible score, and 5 the highest) under the categories of "easiness", "helpfulness" and "clarity."' Additionally students may fill out an "interest" field in which they indicate how appealing the class was before enrolling, and a 350 character summary of their class experience. We focus on perceived clarity because of the direct link between clarity and quality of instruction.

We first select a random sample of 50 international universities that teach Computer Science from the Academic Ranking of World Universities [2]. From our sample of 50 univer-

Table 1: Programming Language Statistics

	Value	Std.Err	t-value	Pr<|t|	n
C	3.38	0.32	10.58	0.0000	109
C++	3.30	0.31	10.65	0.0000	214
Java	3.62	0.19	19.33	0.0000	353
Python	3.70	0.26	14.50	0.0000	133
Scheme	4.06	0.47	8.61	0.0000	32
Scratch	3.91	0.84	4.67	0.0000	49

Table 2: Grading Criteria Statistics

	Clarity	Std.Err	t-value	Pr<|t|	n
Exam Heavy	3.23	0.12	26.91	0.000	726
Equal Mix	3.52	0.14	26.04	0.000	484
Exam Proj	3.65	0.13	27.76	0.000	610
Exam HW	3.12	0.13	23.53	0.000	415

sities, 41 universities are English speaking. The nine non-English speaking universities are removed from our sample. We scrape and parse the reviews of the ratings website for all professors within the computer science departments of the universities in our sample. We remove the ratings from faculty that were rated by fewer than 30 students. This narrows our final sample to 10,655 different reviews of 180 different professors teaching 1,112 courses at 22 universities.

We use Amazon Mechanical Turk [1], a crowdsourcing platform, to find course features for each of the courses in our ratings sample. We do this by asking respondents to fill out a survey. The survey requests to find the online syllabus that corresponds to the course and professor from which we have ratings that is closest to the date of the student review we collected.Then, using the syllabus, respondents are asked to to provide the programming language(s) used, the textbook(s) used, and the percentage of the grade that was determined by homework, projects, quizzes, exams.

From our original sample of 1,112 courses taught by a unique professor, respondents find an online syllabus matching the professor for 342 courses (∼31%). We hypothesize three explanations for the missing syllabi: (i) the syllabi may be accessed only with a password through a course management system, such as blackboard, (ii) the syllabi may not be available only, or (iii) the respondents are not able to find the syllabi.

3. LEARNING STUDENT OPINIONS

We make use of the ratings and syllabi data collected to provide insights into which programming languages beginning students associate with clear instruction. We filter the data to only include introductory level courses (one which does not require any prerequisite coursework in computer science). Our restricted sample includes 1024 reviews. We explore the relationship between clarity ratings and programming language using general linear mixed modeling with random professor and course effects. We do not report programming languages with less than 30 student reviews. Table 1 summarizes the perceived clarity of courses by programming language (higher is better). An intercept is not modeled in order to make the results easily interpretable. The mean clarity rating for introductory courses is 3.599.

We found C and C++ had the lowest coefficients (i.e. compiled languages were less clear). Observe that Scheme and Scratch have the highest clarity ratings followed by Python and Java. We note that the standard errors are smallest for Java and Python and largest for Scheme and Scratch. There is more variation in reviews of courses using Scheme and Scratch than there is for courses using Java and Python. Students in our sample associate clearer instruction with interpreted languages rather than compiled languages.

To assess students' course ratings of clarity based on the percentage of the grade due to exams, quizzes, homework and projects, we created a factor made up of four clusters representing four ways of weighting homework, projects, exams, quizzes and miscellaneous (such as extra credit) for the students' grade. We sort the data to only include observations in which the grading criteria is available and sums to 100. There are 2225 observations with full grading criteria. We use k-means clustering to partition the 2225 observations with complete grading criteria information based on the five aforementioned variables. We optimize our number of clusters by examining how the BIC and AIC of the mixture model change based on the number of clusters selected. A four cluster solution optimizes the AIC and log-likelihood of the model. The cluster membership is modeled using random professor and course effects.

The exams and projects cluster has the highest estimate of clarity. We find that weighting projects equally with exams is associated with a clearer course experience. The equal mix cluster also is associated with higher clarity estimates. The exam heavy cluster and the exam and homework heavy clusters are associated with lower student clarity ratings. We find that a rubric that weights exams and projects evenly is correlated with clearest instruction.

4. CONCLUSIONS

We demonstrate how the Analyzing CurrIculum Decisions (ACID) methodology can be used to leverage collective intelligence and learn student opinions. In introductory computer science courses, we find that students that are taught interpreted languages find their classes clearer. We also that find students who are given an even weighting of exams and projects find their classes clearer. Our study does not necessarily suggest that teachers should change their programming language. Further research is needed before drawing causal inferences. Student evaluations often include free form text where students can describe their experience in the course. One extension is to regress text sentiment on course features. ACID is a useful tool to discover patterns in student opinions. Syllabus data and course ratings data are becoming increasingly available on the Web. This data is used by millions of students and worthy of further research.

5. REFERENCES

[1] Amazon. Mechanical turk, 2014.

[2] MultiMediaLLC. Academic ranking of world universities, 2013.

[3] T. Robinson and W. Hope. Teaching in higher education: is there a need for training in pedagogy in graduate degree programs? *Research in Higher Education Journal*, 2013.

[4] J. Swapceinski. Rate my professor, 2014.