

A Preliminary Analysis of the Logged Questions that Students Ask in Introductory Computer Science

Cecily Heiner

cecily@cs.utah.edu

School of Computing, University of Utah

Abstract. Asking questions is widely believed to contribute to student learning, but little is known about the questions that students ask or how to exploit them in tutorial interventions to help students learn. This paper presents a preliminary analysis of student questions from an introductory computer science course, logged automatically when students requested help during open lab consulting hours. The data set consists of one hundred and forty two questions that introductory computer science students asked while working on four weekly programming assignments. The initial data suggest that student questions can be repetitive in nature and different students ask different kinds of questions. The paper concludes by suggesting that an analysis technique that is more sophisticated than cosine similarity applied to raw text with stop words removed will be necessary to classify student initiated questions.

1 Introduction

Some students in introductory computer science classes have had little or no prior exposure to programming, and many of their questions are poorly articulated. Students depend on teaching assistants to provide help and feedback. Unfortunately, teaching assistants for introductory computer science classes are a scarce and expensive resource, and the demand for their attention often exceeds their supply of time. These demands come from students who are in the lab as well as students working remotely who send questions electronically.

Previously, students in Computer Science 1 at the University of Utah who wanted help used a Java applet called the TA Call Queue where they entered their name and the location of their lab machine. This information then appeared in the TA interface, and a human TA was required to walk over to the student's machine in order to help them. The TA Call Queue did not allow students to type natural language describing their help request, help students obtain remote assistance, or log data to the server. To improve upon the TA Call Queue, a new piece of software called the Virtual Teaching Assistant was built that logged student questions and allowed for remote feedback when necessary. With both the TA Call Queue and the Virtual TA software, the human TAs periodically circulate through the lab and prompt the students to ask questions because some students are reluctant to ask questions without prompting.

The Spring 2008 version of the Virtual Teaching Assistant student software works as follows. The student decides to ask a question and launches the Virtual TA software by clicking on a button on the class webpage. The application already knows the student login, location, and current assignment number. The student can fill in the name of the Java method and class, and any natural language they need to express their question.

Additionally, the student must attach their source code folder using a standard open file dialog prior to submitting their question. When the student clicks the send button, the client connects to a server, logs all of the information it has collected from the student, and passes the question to a human TA on duty. The student interface is shown in Figure 1a.

The human TA sees the question, source code, and other educational context in a TA interface. In the TA interface, the upper left hand area displays the list of students who have questions on the queue, the lower left hand area lists the source code files associated with the selected student. The center area displays the student source code. The third panel to the right displays the student form and provides an area for the TA to type a response and/or an answer category. An answer category corresponds to a group of questions that could be answered with the same answer. To facilitate the assignment of questions to answer categories, the TA interface has a button panel in the right-most pane with one button for each answer category for the current assignment; the TA can also add a new answer category if the current question does not fit into an existing category. The human TA can answer the question in person by walking to the student's machine in the computer lab, or the human TA can type a text response to the student that will appear in the student's client software in the lower text area. To remove the question from the queue, the TA must assign the question to an answer category. The answer category chosen by the human TA is logged to a database along with any text that the human TA has provided for the student. The TA interface is shown in Figure 1b.

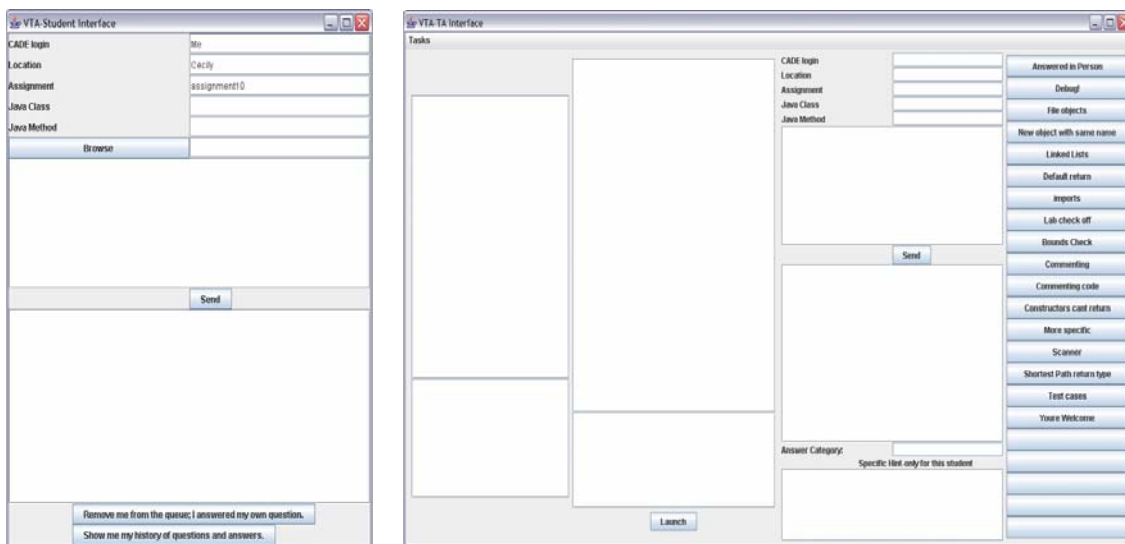


Figure 1. Virtual Teaching Assistant Interfaces for Student(a) and TA(b)

One of the long term goals of the Virtual Teaching Assistant project is to provide automated answers to some of questions that students ask. This paper examines the data logged during one month of usage. The paper presents a preliminary analysis of finding similar questions, and it concludes that applying cosine similarity to raw text with the stop words removed is insufficient for finding similar previous questions that could be exploited in a tutorial intervention.

2 Prior Work

Because novices are notoriously inept at articulating their reasoning, many intelligent tutoring systems use one of two approaches to dealing with the questions that students ask. Several tutoring systems have completely bypassed the problem of poorly articulated student language by restricting the size of the information space that the system covers (e.g. single physics problem[3], a sentence to be read aloud[8], or algebra symbolization[5]), dividing each problem into a set of skills, and creating interventions or help messages for each skill. Alternatively, some systems have focused specifically on helping students to articulate their reasoning (e.g. providing explanations for steps in a geometry problem[2], describing general physics principles in natural language[4], and listing required hardware for a computer task[4]) because generating natural language may help students develop deep learning. However, these systems are generally trying to elicit specific pieces of natural language from the student instead of providing answers to student questions. Research on producing automated responses to student generated questions is remarkably sparse throughout the literature.

3 Study Design

3.1 Participants

The study participants were students from Introduction to Computer Science 1 (CS1410) at the University of Utah. Most students in Computer Science 1 are age 18-22, but there are also a few non-traditional students, such as a student from a local high school or adults who have returned to school later in life. Computer Science 1 is the first required computer science course for computer science majors, and it is a gatekeeper course with a strong emphasis on the Java programming language as well as the traditionally long hours for novice programmers and the typically high dropout, fail, and withdrawal rates. The majority of students who take Computer Science 1 hope to major in computer science or a related field, but they must pass that class along with three others with sufficiently high grades to attain official status as a computer science major. Although approximately seventy eight students were active in the course during the study, only twenty four of them asked questions using the Virtual Teaching Assistant software during the month long study period.

3.2 Data Cleansing

When the data logging software was designed, every effort was made to facilitate rapid data analysis. The long term goal is to complete the analysis for a question in real time and exploit it for an instant tutorial intervention. However, even though the data set was carefully designed, some cleansing was necessary. For example, some students used several computers away from the lab, and their logins needed to be recoded for consistency. In a few cases, the only way to obtain their login was to look in the comments in the source code. Some students asked the same question twice because of a glitch in the software that caused a delay between question submission and system acknowledgement; the duplicate questions were removed from the dataset, but the original questions were left in the dataset. If the human TA did not provide an adequate

answer category for a student question, and a category could not be generated using the student’s natural language or the status of the source code, then the question was excluded.

4 Similarity Analyses

To calculate the similarity of a new question to a previous questions, the natural language from the student questions is extracted and represented in vector form where each row of the vector corresponds to a particular word, and the value of that vector element is the number of times that word appears in the student’s question. The vector representation excluded stop words from this list[1]; stop words are words such as “the”, “a”, “I”, “you”, and other common words. A vector similarity measurement calculates the similarity of a pair of questions on a scale of 0 to 1. In these measurements, 0 means that the pair has no common words, and 1 means that the questions are identical. The experiments described below utilize cosine similarity[7].

As a gold standard, a question is similar to a previous question if it has the same answer category. Table 1 provides an illustrative data set. To conserve space, questions and answer categories are referred to by number; in the actual analysis, the original natural language is used. As described in the introduction, when using the Virtual TA system, the human TAs must assign an answer category to each student question that they answer. A previous question with the same answer category is called a *target question*. In Table 2

Table , “target question(s)” lists previous questions with the same answer category. Additionally, for each question, the similarity between that question and each previous question is calculated. The previous question that has the highest similarity score when paired with the current question has the “max similarity score”, and it is considered the “most similar previous question”. If multiple previous questions have the same highest similarity score, the most recent question is considered the “most similar previous question”.

Table 1. Question Similarity Scores

	Answer Category	Target Question(s)	Q1 Sim.	Q2 Sim.	Q3 Sim	Q4 Sim	Q5 Sim	Q6 Sim	Max Similarity Score	Most Similar Previous Question
Q1	A1									
Q2	A2		0						0	Q1
Q3	A3		0	0					0	Q2
Q4	A1	Q1	0.47	0	0				0.47	Q1
Q5	A4		0	0	0	0			0	Q4
Q6	A4	Q5	0	0.03	0	0.12	0.1		0.12	Q4
Q7	A5		0.1	0.05	0.05	0.04	0	0	.05	Q3

The original plan was to simply use the answer categories as recorded by the human TAs. Unfortunately, not all of the answer categories that the TAs chose are particularly descriptive of the student’s question. For example, the human TAs often used the category “Answered in Person”, as opposed to a category that provided a meaningful description of the student’s question. Excluding such poorly described answer category data results in a smaller dataset as shown in the “Excluding recoded data” and “Excluding both” columns of Table 2. A second alternative is to recode the data; resulting in a larger dataset as shown in the “Everything” and “Excluding compiler errors” columns of Table 2. This disaggregation shows how much of each type of data was recoded.

An additional disaggregation considers the questions associated with source code that doesn’t compile. Because the compiler can indicate without human intervention whether or not the source code compiles, the kinds of automated data collection and analyses that are possible differ depending on whether or not the source code compiles. For example, if the source code does not compile, the compiler output may be exploited for question classification. If the source code does compile, then comparison algorithms can exploit the output of parse tree analysis or runtime analysis techniques such as program slicing and comparison[6].

Table 2. Data set with and without exclusions

	Everything	Excluding recoded data	Excluding compiler errors	Excluding both
Total Questions	142	87	80	53
Target Question Exists	35	13	11	6
Target Question Found	6	3	4	2

In all four conditions shown in Table 2, the algorithm found similar previous questions for at least a few questions, but only a very small percentage of the time. Similar previous questions could be exploited in a tutorial intervention if a similarity score threshold separates those questions with a target question from those questions without a target question. Figure 2 is an attempt to find such a threshold when excluding compiler errors. The independent variable is the maximum similarity score for a question, and the dependent variable 0 if there was not a similar previous question, 1 if there was a most similar question and it was found by the algorithm, and 0.5 if there was a similar previous question, but it was not found by the algorithm. If Figure 2 were a step function, then the threshold would be the location of the step. Unfortunately, this data does not seem to have such a threshold. Consequently, applying cosine similarity to raw text with stop words removed is insufficient to classify many student questions, and maximum similarity score with natural language is insufficient to identify previous questions to exploit in an automated tutoring intervention.

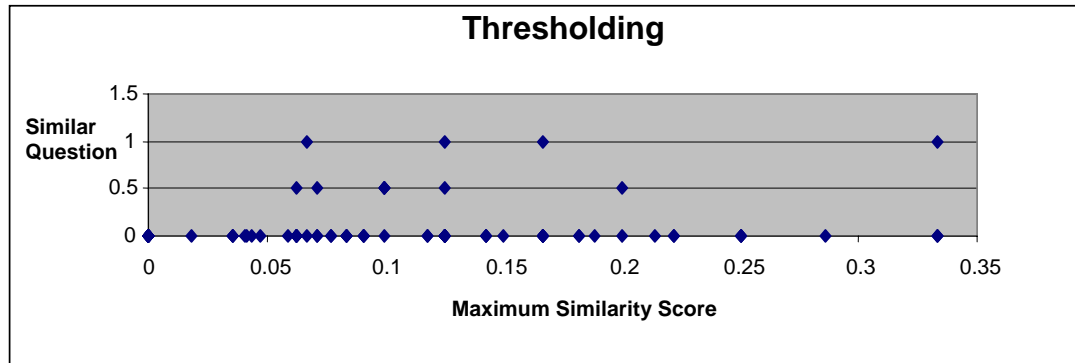


Figure 2. Thresholding

4 Conclusions

4.1 Limitations

This work has all of the standard limitations of work based on a single intelligent tutoring system deployed in a single class; the results hold for this population of students using this tutoring system in the setting described in the paper. Additionally, the numbers are small because it is a only month long study. The paper does not show p-values for any of the claims, nor does it calculate the correlations of variables. A first attempt at correlating the number of questions with average assignment score or midterm, not presented in this paper, suggests that this is a difficult problem for this domain because several of the students who ask questions early in the semester decide to drop, withdraw, or fail before the first midterm. How to properly deal with such students is an interesting, open research question for educational data mining.

4.2 Contributions

This is the first paper to analyze data from the Spring 2008 version of the Virtual Teaching Assistant, a tool that logs the questions that students ask and the answer categories that human TAs use to describe student questions. A set of preliminary analyses suggest that using raw natural language is not sufficient to find previous student questions that can be exploited in an automated tutorial intervention.

4.3 Future Work

This paper presents a preliminary analysis of a month-long study of student questions using only the natural language of the questions that students asked. This analysis may be refined using a more complete task-specific stopwords list. Several words such as “cool”, “awesome”, “thanks”, “need”, “help”, and “question” that are not normally considered stopwords behave like stop words for this task. Treating them like stopwords may improve accuracy. Additionally, a more vigorous recoding of answer categories and/or a clearer set of guidelines for creating answer may improve question classification accuracy. Even in the recoded categories, there are several redundant answer categories

such as “Moving the pyramid”, “sizing the pyramid”, and “Pyramid location” that should probably all be recoded to the same general category; another example is the redundant answer categories of “mortgage equation” and “calculating mortgage”.

Previous work suggests that the majority of questions asked by students about a programming assignment are often clustered around one or two topics[9], but these topics are too vague to be useful for question classification; creating a question classification scheme that is simple enough for introductory computer science TAs to use is an area that merits further research.

The next step for this research is to cleanse the data more thoroughly and scale up this study to a semester long study, and include other forms of data such as student source code and educational context, e.g. the assignment the student is working on, as part of the classification algorithm. The plan is to compare the plain natural language version with the extended version that uses source code and educational context, and hopefully show that using the extra data results in a statistically significant improvement in question classification. Also, with improved question classification, it may be possible to exploit the question classification in a tutorial intervention to help students learn more. Showing that previous questions, mined in real time from the knowledge base of the system, can be used in a tutorial intervention to help students learn is the long term goal of this research project.

In addition to extending the VTA project, future work is needed to clarify the differences in student and tutor initiated open ended dialog. Previous work has shown that cosine similarity was as effective as an untrained human tutor in a tutor initiated open ended dialog[10], but in this scenario, the student can exploit both the natural language in the tutor-initiated question as well as text within a window from which the question was drawn. Consider the example given in that paper: The tutor-initiated question is “How is an operating system like a *communications* coordinator?”, a student response is “It *communicates* with the peripherals”, and a model answer is “A computer's operating system includes programs that take care of the details of *communication* with peripherals.” (emphasis added) As the emphasis added shows, many terms that are used in a tutor-initiated question also tend to appear in both the student answer and the model answer. The topic of this question was probably drawn from a specific segment of a written text, and with high probability the student would have extracted their answers from a segment of the text with the same information. Student initiated questions do not have the added advantage of the language from the tutor-prompt, and it is not clear whether or not cosine-similarity can be exploited to help answer them. A more detailed comparison of several dialogues from both student-initiated and tutor-initiated systems may shed light on additional differences in dialogues based on who initiates them.

Acknowledgements

Thanks to Hal Daume, Joe Zachary, Joseph Beck, Peter Jensen and others who have provided helpful comments and feedback on various versions of this work.

References

- [1] cited 2008; Available from:
http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words.
- [2] Aleven, V., Popescu, O., Koedinger, K. Pilot-Testing a Tutorial Dialogue System That Supports Self-Explanation. *Intelligent Tutoring Systems*, 2002. Springer Berlin / Heidelberg.
- [3] Gertner, A. S., VanLehn, K. Andes: A Coached Problem Solving Environment for Physics. *Intelligent Tutoring Systems*, 2000. Springer.
- [4] Graesser, A. C., VanLehn, K., Rosé, C. P., Jordan, P. W., Harter, D., Intelligent Tutoring Systems with Conversational Dialogue, in *AI Magazine*. 2001. p. 39-50.
- [5] Heffernan, N., Intelligent Tutoring Systems have Forgotten the Tutor: Adding a Cognitive Model of Human Tutors, in PhD Thesis. 2001, Carnegie Mellon University: Pittsburgh, Pennsylvania.
- [6] Jensen, P., Hybrid Automated Fault Localization in Programs written by Novice Programmers, in PhD Thesis. 2007, University of Utah: Salt Lake City.
- [7] Manning, C. D., Schütze, H., *Foundations of Statistical Natural Language Processing*. 1999, Cambridge, Massachusetts: The MIT Press.
- [8] Mostow, J., Aist, G., Evaluating tutors that listen: An overview of Project LISTEN, in *Smart Machines in Education*, K. Forbus and P. Feltovich, Editors. 2001, MIT/AAAI Press: Menlo Park, CA. p. 169-234.
- [9] Robins, A., Haden, P., Garner, S. Problem Distributions in a CS1 Course. *Australian Computing Education Conference*, 2005. Conferences in Research in Practice in Information Technology.
- [10] Wiemer-Hastings, P., Wiemer-Hastings, K., Graesser, A. C. Approximate natural language understanding for an intelligent tutor. *12th International Florida Artificial Intelligence Research Conference*, 1999. AAAI Press.