Ogata, H. et al. (Eds.) (2015). Proceedings of the 23rd International Conference on Computers in Education. China: Asia-Pacific Society for Computers in Education

Predicting Quitting Behavior in SQL-Tutor

Jin Kwang HONG, Antonija MITROVIC* & Kourosh NESHATIAN
Department of Computer Science and Software Engineering, University of Canterbury, New Zealand
*tanja.mitrovic@canterbury.ac.nz

Abstract: Although Intelligent Tutoring Systems (ITSs) have proven to be very effective in supporting learning, keeping students who interact with them engaged in their activity remains a challenge. In this study, we use machine learning techniques to predict whether the student is going to abandon the current problem. The study has been done in the context of SQL-Tutor, a constraint-based ITS that teaches students how to query relational databases. We extracted a number of features from past data and used the J48 algorithm to train a decision tree. The model was used in a lab session to make predictions and provide limited intervention in order to prevent potential abandonments. Overall, the classifier demonstrated a promising performance. The results also provided insights as to what areas can be improved in future.

Keywords: Quitting behaviour, disengagement identification and repair, educational data mining

1. Introduction

Intelligent Tutoring Systems generally leave a lot of control over learning in the student's hands. Many ITSs allow students to select problems themselves, to decide when to move to a new problem, and in some cases when and how much feedback to require from the system. Although this freedom is used wisely by some students, not all of them possess appropriate self-regulation skills and, as the result, do not learn effectively.

One of the major problems identified in students' interaction with ITSs is disengagement, which appears in different ways in ITSs. A lot of research has been done on detecting when student game the system, a set of behaviours in which students exploit various strategies to get solutions or specific problem-solving steps from the ITS rather than solving problems. Gaming the system includes systematic guessing and help abuse. This kind of behaviour results in suboptimal learning, with some studies showing that students who game learn two thirds of what students who do not game learn (Baker, Corbett & Koedinger, 2004). Detectors of gaming the system have been developed through both educational data mining/machine learning methods (Baker et al., 2008; Baker, Mitrovic & Mathews, 2010; Beal, Qu & Lee. 2008; Walonoski & Heffernan, 2006), and through knowledge engineering approaches (Aleven et al., 2006; Beck, 2005). These detectors have been incorporated into interventions which were shown to reduce gaming and improve learning. An example intervention was based on an animated agent which showed negative emotions when students gamed the system (Baker et al., 2006). Other interventions that deal with disengagement involved meta-cognitive feedback, which supports the student in thinking about their unproductive behaviour and illustrate successful behaviours. An example of this is the use of simple visualizations of the student's progress between problems (Arroyo et al., 2007), or using open learner models as a tool to involve students in reflection on their knowledge (Mitrovic & Martin, 2007). There are also positive results achieved when detecting disengagement from eye gaze data, and using an animated agent to re-direct the student's attention to important areas of the interface (D'Mello et al., 2012). Other types of disengagement include off-task behaviour, boredom and carelessness (D'Mello et al., 2012; Drummond & Litman, 2010).

Mills and colleagues (2014) report on predicting quitting behaviour in students learning from instructional texts. Their predictor generates a probability that a student would quit an upcoming instructional text, based on a number of features identified from reading previous texts. Quitting was predicted with the accuracy of 76.5%. The predictor also generates probabilities for quitting on the first page or later in the given text.

In this project, we focus on a particular type of disengagement: we are interested in predicting whether a student is going to abandon the current problem without completing it. This kind of behaviour might be the result of disengagement and low motivation. Our goal is to investigate whether it is possible to predict such behaviour, so that an intervention can be developed for the ITS to try to motivate the student to persevere. We developed a simple intervention in such cases, providing motivational messages to students, while at the same time collecting subjective data from the students about the accuracy of predictions.

In Section 2, we briefly introduce SQL-Tutor, the ITS which is the context of our study. Section 3 presents the information about the data set used to generate the predictors and select the one to be used in the study. The details of the study are presented in Section 3, followed by the discussion of the results in Section 4. We conclude with presenting several avenues for future work.

2. SQL-Tutor

SQL-Tutor is a constraint-based tutor that teaches SQL, the dominant query language for relational databases (Mitrovic, 1998; 2003). It is a mature ITS, designed as a practice environment with the prerequisite that students be previously exposed to the SQL concepts in lectures. Students submit solutions which are sent to the student modeller for analysis. The student modeller identifies any errors and updates the student model accordingly to reflect the student's progress within the domain. To check the correctness of the student's solution, SQL-Tutor evaluates the student's solution against domain knowledge, represented as a set of 700 constraints. A preferred solution, specified by the instructor, aids in this evaluation process while still allowing for novel correct solutions from the student. A constraint consists of a relevance condition C_r, which checks whether the constraint is appropriate for a particular student's solution, and a satisfaction condition, C_s. A solution is correct if it satisfies the satisfaction conditions of all relevant constraints.

Once the student's solution is evaluated, the student model passes information to the pedagogical module which generates the appropriate feedback. If any constraints are violated, SQL-tutor will provide feedback on them. In the case where the solution is correct or the student requires a new problem to work on, the pedagogical module uses the information from the student model to select an appropriate problem.

SQL-Tutor provides feedback on demand only, when the student submits the solution. The system offers six levels of feedback, differing in the amount of detail provided. On the first attempt, the system only informs the student whether the solution is correct or not. The second level (*Error Flag*) points to the part of the solution that is incorrect. The third level (*Hint*) provides a description of one error, pointing out where exactly the error is, what constitutes the error (performing blame allocation) and referring the student to the underlying domain principle that is violated (revising student's knowledge). The hint message comes directly from the violated constraint. The automatic progression of feedback levels ends at the hint level; to obtain higher levels of feedback, the student needs to explicitly request them. For example, the student can ask for the hint message for all violated constraints (*All Errors*), a partial solution (showing the correct version of one part of the solution that is wrong), or a complete solution for the problem.

Figure 1 presents the interface of the standard version of SQL-Tutor. The interface presents the text of the problem at the top, the area for the student to enter his/her solution on the left, and the feedback area on the right. The bottom part of the interface presents the schema of the currently selected database, from which the student can get additional information about the tables, their attributes and can view the data.

SQL-Tutor has been in regular use in databases courses since 1998. Over the years, we have conducted 17 evaluation studies at the University of Canterbury alone, with additional studies being done at other universities. Furthermore, SQL-Tutor has been accessible via the Addison-Wesley's DatabasePlace Web portal since January 2003. In previous work (Baker, Mitrovic & Mathews, 2010), we developed a gaming detector, but the current version of SQL-Tutor does not provide any support for disengaged students.

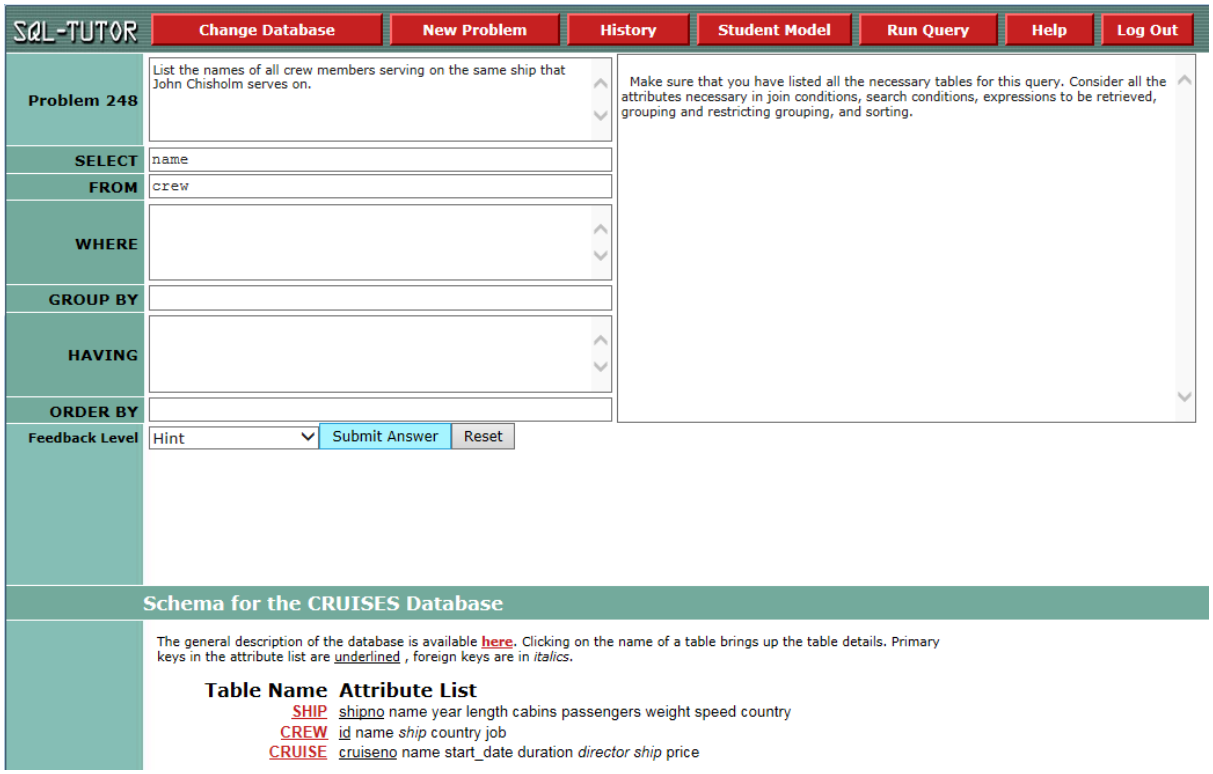

Figure 1. A screenshot of SQL-Tutor

3. Building a Predictive Model

SQL-Tutor maintains a log file for each student, containing information about each session and all actions the student performed. We used this information to predict weather student is going to abandon the current problem. In the following subsections, we provide details about the data sets used, the features we extracted from the logs, and the classifier generated from the features.

3.1 Data Sets

We used two data sets: the DatabasePlace dataset, consisting of 7,519 logs (collected from the portal from 2003 to 2007), and the 2010 dataset, collected from 56 students taking the course on relational databases at the University of Canterbury in 2010. The two datasets do not only differ in their sizes: we know much more about the 2010 dataset, as the students were local students taking a course taught by the second author of this paper. Those students attended lectures and labs on SQL, and used SQL-Tutor as an additional tool to support their learning whenever they wanted. On the other hand, we know nothing about the backgrounds of the students using SQL-Tutor on DatabasePlace: students could have bought access cards online or together with one of the database textbooks published by Addison-Wesley. Those students may have had very different background knowledge, and were located worldwide. Additionally, we have no information about how they used SQL-Tutor, whether as a part of a course, or completely independently. The only information available for the DatabasePlace data is contained in system logs.

The granularity level of interest for our study is an *attempt*, which is a solution that the student submits for a problem. Typically students make more than one attempt per problem. Table 1 reports the number of instances in each dataset. The *Abandoned* column presents the number of instances which resulted in the student abandoning the current problem. The percentage of abandoned instances is much higher for the DatabasePlace dataset (47.5%) in comparison to the 2010 dataset (11.5%), which illustrates the difference in the two student populations.

Table 1: Information about the datasets used

Dataset	Logs	Instances	Abandoned
DatabasePlace	7,519	16,541	7,854
2010	56	1,980	227

3.2 Extracted Features

The first step is to distil a set of features to describe each instance (i.e. attempt), which could potentially be predictive of the quitting behaviour. Many main-stream machine learning algorithms such as decision trees, (non-recurrent) neural networks, SVMs, etc. are not sensitive to the order in which the instances are presented; the input vectors are assumed to be independent of each other. On the other hand, in this study we would like to have models that take recent actions of students into account as they happen and update their prediction. A common technique to tackle this problem is to extract features from a moving time window and train a classifier that can make good predictions based on cumulative information about past events. Since we were interested in students' actions leading up to quitting a problem, we identified three categories of features: current problem features, previous problem features, and cumulative features representing all previous problems in the session.

The current problem features provide information about the current problem the student is working on, and can potentially abandon. This category includes 10 features: the numbers of violated and satisfied constraints, problem complexity, the level of feedback received for the attempt, attempt number, elapsed time from the beginning of the current problem, time since previous submission, time since session start, the squared difference in times for the current and the previous submission, and a Boolean feature specifying whether the number of mistakes is smaller in the current submission compared to that of the previous one.

There were 16 features representing the previous problem: whether or not the problem was completed, the first submit time, time taken to complete the problem, number of attempts, the maximum number of violated constraints, the number of attempts with mistakes, average attempt time, the time for the last attempt, –standard deviation of submit times, maximum/minimum submit time, problem complexity, a Boolean feature representing whether the previous problem is from the same database as the current problem, the number of distinct feedback levels the student received for the previous problem, the number of identical solutions submitted, and the total time spent on the problem.

There were 24 cumulative features representing the current session, which include the numbers of attempted/completed problems, the average, standard deviation and maximum problem complexity, the average and standard deviation of the number of attempts per problem, and the number of times the student changes databases. Additionally, this category includes the average, standard deviation, minimum and maximum for the first submission time, time between submissions, the number of errors, and completion time.

There was also a class feature, specifying whether a particular instance represented an abandonment or not. Therefore we had a total of 51 features representing attempts.

3.3 Classifier

We chose decision trees for our classification model. One reason for this choice is the presence of categorical features. Decision trees are generally much better at handling categorical features than discriminative approaches (such as canonical neural networks) where the categorical values must be converted to (often arbitrary) numeric values. The other reason for choosing decision trees is that if something goes wrong, they are easier to investigate.

We used decision tree learning algorithms provided in WEKA (Hall et al., 2009). The data was transformed to the ARFF format and processed in WEKA. We experimented with a number of different algorithms, and obtained the best results with J48 decision trees (Quinlan, 1993). When J48 was applied to the 2010 dataset using 10-fold cross-validation, the predictor correctly classified 97.12% of instances. When applied to the DatabasePlace dataset, 89.54% of the instances were classified correctly. The 2010 dataset, as described previously, was collected from students taking a course at the University of Canterbury, similar to the context in which the current study was performed. However, the 2010 dataset

was small, containing only 56 logs, which is not enough to produce a good classifier. For that reason, we decided to combine the two datasets.

Table 2 shows the performance of five learning algorithms on the combined data set. Only algorithms that are capable of handling both numerical and categorical features can be used with the data set. The true-positive rate (TPR) is the fraction of positive (abandoned) instances that have been classified correctly, and the false-positive rate (FPR) is the fraction of negative (not-abandoned) instances that have been incorrectly labelled as positive (abandoned). The accuracy is the ratio of the number correctly-classified instances to the total number of instances. The reported measures are average values obtained by 10-fold cross validation.

Table 2: Performance of various learning schemes on the combined data set

	TPR	**FPR**	**Accuracy**
Decision Trees (J48)	0.88	0.09	89.73%
Bayesian Networks (K2)	0.58	0.08	77.44%
Logistic Classification	0.71	0.13	79.67%
AdaBoost with Decision Stumps	0.57	0.10	75.56%
Classification/Regression Trees (M5P)	0.85	0.10	87.80%

The best performing predictor in terms of accuracy and TPR, is produced by J48 decision tree algorithm with an accuracy of 89.73%. The FPR of the J48 decision tree predictor is 1% higher than the predictor produced using Bayesian networks but considering the overall performance and that predicting cases that are going to abandon is more important in our application, we chose the decision tree model as our classifier.

Table 3 shows the confusion matrix of the classifier. The values are aggregates of values obtained during a 10-fold cross-validation. That is, the data is divided into 10 stratified partitions (with the same class ratios). Each time, one partition (fold) is used for testing and the remaining nine for building (training) a model. The confusion matrix is initialized to zero and is accumulated over time based on the performance of the built model on the test partition. Note that the confusion matrix in Table 3 yields the performance measures reported in the first row of Table 2.

Table 3: The confusion matrix for the classifier

		Predicted	
		Abandoned	Not abandoned
Actual	Abandoned	7,124	957
	Not abandoned	945	9,495

Finally we built a decision tree based on the entire data set to use later for prediction and also gain some insight into the structure of the built tree. The decision tree had 1,565 nodes, out of which 783 were leaf nodes (class labels). This size of tree is considered somewhat large compared to typical decision trees. There are several reasons for this. Out of 50 input features, 45 features were part of the built tree. Many of these features are numeric and thus can be used multiple times across the path from the root of the tree. In fact, any non-rectangular decision region would require the algorithm to condition on a feature multiple times (Quinlan 1993). Another reason for the large number of nodes is the presence of redundant subtrees which are intrinsic of decision tree representation and are also sometimes a consequence of the greedy nature of the learning algorithm. If having a small and intelligible decision tree is desired, then much smaller trees can be produced by limiting the depth of the decision tree during training or by using only a small subset of extracted features. For example, the decision tree constructed using only one feature (the number of violated constraints, which is equivalent to the number of errors) contains only a root node and two leaves, with the accuracy of 71%.

4. Study

The study was conducted in a regular course laboratory session (100 minutes long), with 49 students enrolled in the relational database course at the University of Canterbury. We obtained the approval for

the study from the Human Ethics committee of the University of Canterbury. Prior to the study, the students have attended five lectures and two labs on SQL, but they encountered the ITS for the first time in this study. The students first gave written consent, and then started interacting with SQL-Tutor. The data collected about three participants were removed from analyses, as they have used SQL-Tutor for less than 10 minutes. The majority of participants were males of ages 18–23, and only 7 females.

As stated earlier, one of the goals of our project was to investigate whether it is possible to predict when the student is going to abandon the current problem. We also wanted to investigate whether a simple intervention can be effective in reducing the number of abandoned problems. We have modified SQL-Tutor by adding a component that calculates the feature values as the student is solving problems. For each submission, the classifier produced a prediction, which was stored in the log. This modification enables us to study the accuracy of the predictor.

We also developed a simple intervention for situations when the prediction is that the student will abandon a problem. In such a case, the system shows a short motivational message to the student, encouraging him/her to continue working on the problem. We defined 11 messages, such as *With effort, you can master SQL*. The message to be shown was selected randomly, and was shown to the student in a pop-up window three seconds after the submission. The time delay was selected so that the student has time to observe the feedback on the submitted solution. In addition, the student was asked to specify whether he/she was about to abandon the problem or not. This feature allowed us to collect feedback from students on the accuracy of the predictor.

The session was divided into two phases. During the first phase (40 minutes long), SQL-Tutor was making predictions in the background without any visible changes shown to the participants. In the second phase (60 minutes), the students received intervention messages when the prediction was that they would abandon problems. At the end of the session, the participants completed questionnaires.

5. Results

The participants attempted 9.94 problems on average in Phase 1, ranging from 4 to 19, and completed 9.09 problems on average (Table 4). The number of submissions ranged from 7 to 128, with the average of 40.63. Some participants left the study during Phase 1, and the average time spent in Phase 1 was 35.61 minutes (sd=10.92). In Phase 2, we had 38 participants who spent 26.14 minutes on average.

Table 4: Statistics about the two phases

	Phase 1	**Phase 2**
Duration (minutes)	35.61 (10.92)	26.14 (14.57)
Attempted problems	9.94 (4.16)	5.29 (3.56)
Completed problems	9.09 (4.16)	4.63 (3.66)
# Submissions	40.63 (21.74)	22.76 (13.42)
Problem completion rate (%)	90.80 (10.32)	82.82 (24.25)
Problem complexity	2.51 (0.26)	3.77 (0.14)
Abandon predictions	4.07 (5.99)	4.08 (4.65)
Non-abandon predictions	36.59 (15.52)	18.68 (11.21)

After the study, we evaluated the accuracy of the predictor. Table 5 presents the confusion matrix for Phase 1 only, where there were 1,065 submissions. The predictor correctly classified 80% of the submissions. Out of 86 abandoned instances, 24 were predicted correctly giving the true positive rate of 0.28. The not-abandoned cases were predicted correctly at a higher rate, resulting in the true negative rate of 0.85.

Table 5: Accuracy of the predictor in Phase 1

		Predicted	
		Abandoned	Not abandoned
Actual	Abandoned	24	62
	Not abandoned	151	828

The mean problem completion rate was 90.8% in Phase 1, and 82.82% in Phase 2. Although the completion rate was lower in the second phase, the difference is not significant. The lower problem completion rate for phase 2 is the result of the participants attempting more complex problems as the time progressed. Complexity levels of problems in SQL-Tutor range from 1 (the simplest problems) to 9 (the most complex problems). Figure 2 shows the percentages of problems at various complexity levels during the two phases. The average level of problem complexity attempted in Phase 1 is 2.51, compared to 3.77 for Phase 2. In Phase 2, students attempted more difficult problems.

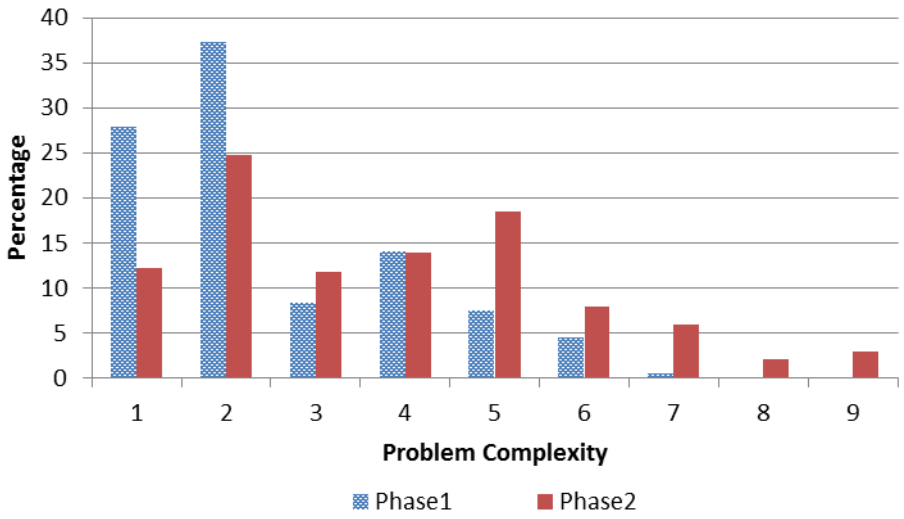

Figure 2. Percentages of problems at various complexity levels

For 14 participants, the classifier made no predictions of abandonment in Phase 2. The accuracy of the predictor for those participants was 99.35% (sd = 2.4%). Those participants completed almost all problems they attempted in Phase 2, with the completion rate of 82.86%.

The remaining 24 participants received at least one intervention, with the average of 6.29 (sd = 4.05), with the maximum of 13 interventions for one participant. The accuracy of the classifier for those participants was 71.17% (sd = 17%). The participants had the opportunity to give subjective feedback when they received interventions. The percentage of participants who agreed with the intervention (i.e. who specified they were going to abandon the current problem) was 20.2%, meaning that participants found the majority of interventions as not appropriate.

The user questionnaire provided additional information about the participants' impressions about the intervention. Forty four participants completed the questionnaire. Eight of those participants have only completed Phase 1, and therefore could not observe the intervention. Additionally, 12 participants who completed both phases and filled the questionnaire have not seen the intervention as the classifier did not produce any predictions that they would abandon problems. Therefore, their responses to the questionnaire are not relevant. It is interesting to note, though, that out of those 20 participants, the majority (13 participants) found the idea of the intervention desirable for increasing motivation and only 2 participants did not like the idea (5 participants provided no answer for that question).

Table 6: Replies from the questionnaire

	No	Yes
Going to abandon?	18	2
Message effective	15	8
Intervention desirable	2	16

Table 6 presents the replies on three questions for the remaining 24 participants who have seen the intervention at least once during Phase 2. Please note that the sum of the numbers in a row is not 24, as not all students answered all questions. When asked whether they were going to abandon the current problem at the time when they received interventions, all but two participants answered negatively. The

participants were also asked if the intervention messages were effective in keeping them engaged, to which 8 participants answered positively and 15 negatively. Those who answered positively provided further comments, saying that even though the intervention was given at the wrong time, it actually encouraged them to solve the problem. One participants noted "*I did not want the system to think I was weak!*" Several students also noted that they received interventions at the time when they were struggling with the problem and were frustrated, but that actually motivated them even more to complete the problem.

The last question asked whether having an intervention is a good idea to keep students engaged. The majority of the students (16) agreed with this statement, and only 2 disagreed. Their comments showed that the participants generally liked the idea of interventions, but warned against providing them too often (i.e. making the predictions more accurate). Some participants found pop-ups annoying.

6. Conclusions and Future Work

This paper presented a study in which SQL-Tutor was extended to be able to predict whether a student would abandon a problem. We developed a predictor based on two datasets: the DatabasePlace dataset, a large dataset for which we know nothing about the student backgrounds, and the 2010 dataset, collected from students enrolled in a database course taught by the second author. The overall accuracy of the predictor on the combined data was high. We then implemented a simple intervention where a message is displayed to the student when the prediction is that he/she would abandon the problem, motivating the student to persevere with problem solving.

We conducted a study in September 2014, to test the accuracy of the predictor and the effect of the intervention. The actual accuracy of the predictor was 80%, but the problem was the low true positive rate. We believe the reason for this was the difference between the 2010/2014 data compared to the DatabasePlace dataset. For the two datasets collected from the University of Canterbury students, the rate of abandoning problems was 8-11%, which is very different from the rate for the DatabasePlace dataset (47.5%). This is one of the limitations of our study. We plan to continue this project, using the 2010/2014 data sets to produce a new classifier and then test it with a new class of students taking the database course.

The replies from the questionnaire show that the majority of the participants (29 out of 44) believed the ITS should intervene when problems are abandoned. Although they noted the interventions were often given at the wrong time, they still found the messages motivational and were challenged further to complete problems.

There are several other limitations of our work. The intervention was only applied in Phase 2 of the study, when there were some fatigue effects, as can be seen from the fact that some students finished the study before the end of the session. Therefore, it would be desirable to study the effect of the designed intervention during the whole session, to determine its true effect. Furthermore, the intervention was very simple, and we plan to improve it for the future studies.

The study was done with a small population of students, and we plan to conduct further studies with more participants. Additionally, we used a single model to predict abandonment for all students. Having user-specific models might help produce a better predictive performance. However, this would require a lot more data; for each user we must have enough past observations to be able to construct individual models.

References

Aleven, V., McLaren, B., Roll, I., & Koedinger, K. (2006) Toward meta-cognitive tutoring: A model of help seeking with a Cognitive Tutor. *Artificial Intelligence and Education, 16*, 101-128

Arroyo, I., Ferguson, K., Johns, J., Dragon, T., Meheranian, H., Fisher, D., Barto, A., Mahadevan, S., & Woolf, B.P. (2007) Repairing Disengagement with Non-Invasive Interventions. In: *Proc. 13th Int. Conf. Artificial Intelligence in Education* (pp. 195-202).

Baker, R. S., Corbett, A. T., & Koedinger, K. R. (2004). Detecting student misuse of intelligent tutoring systems. In *Proc. Intelligent tutoring systems* (pp. 531-540). Springer Berlin Heidelberg.

Baker, R.S.J.d., Corbett, A.T., Koedinger, K.R., Evenson, S.E., Roll, I., Wagner, A.Z., Naim, M., Raspat, J., Baker, D.J., & Beck, J. (2006) Adapting to When Students Game an Intelligent Tutoring System. In: *Proc. 8th Int. Conf. Intelligent Tutoring Systems* (pp. 392-401).

Baker, R.S.J.d., Corbett, A.T., Roll, I., & Koedinger, K.R. (2008) Developing a Generalizable Detector of When Students Game the System. *User Modeling and User-Adapted Interaction*, *18*(3), 287-314.

Baker, R., Mitrovic., A., & Mathews, M. (2010) Detecting Gaming the System in Constraint-based Tutors. P. De Bra, A. Kobsa, and D. Chin (Eds.): *User Modeling, Adaptation, and Personalization* (pp. 267-278). Springer Berlin Heidelberg.

Beal, C.R., Qu, L., & Lee, H. (2008) Mathematics motivation and achievement as predictors of high school students' guessing and help-seeking with instructional software. *Journal of Computer Assisted Learning*, *24*(6), 507-514.

Beck, J. (2005) Engagement tracing: using response times to model student disengagement. In: *Proc.12th Int. Conf. on Artificial Intelligence in Education* (pp. 88-95).

D'Mello, S., Olney, A., Williams, C., Hays, P. (2012) Gaze tutor: A gaze-reactive intelligent tutoring system. *Human-Computer Studies*, *70*(5), 377-398.

Drummond, J., Litman, D. (2010) In the zone: towards detecting student zoning out using supervised machine learning. In: Aleven, V., Kay, J., Mostow, J. (Eds.), *Proc. 10th Int. Conf. Intelligent Tutoring Systems* (pp. 306–308).

Hall, M., Frank, E. Holmes, G., Pfahringer B, Reutemann, P., Witten, I. H. (2009) The WEKA Data Mining Software: An Update*; SIGKDD Explorations*, *11*(1).

Mills, C., Bosch, N., Graesser, A., D'Mello, S. (2014) To Quit or Not to Quit: Predicting Future Behavioral Disengagement from Reading Patterns. In *Proc. 12th Int. Conf. Intelligent Tutoring Systems* (pp. 19-28), Springer.

Mitrovic, A. (1998) Experiences in implementing constraint-based modeling in SQL-Tutor. In B. Goettl, H. Halff, C. Redfield, & V. Shute (Eds.), *Proc. Int. Conf. on Intelligent Tutoring Systems* (pp. 414–423).

Mitrovic, A. (2003) An Intelligent SQL Tutor on the Web. *Artificial Intelligence in Education*, *13*, 173-197.

Mitrovic, A., & Martin, B. (2007). Evaluating the effect of open student models on self-assessment. *Artificial Intelligence in Education*, *17*(2), 121-144.

Quinlan, J. R. (1993) *C4. 5: programs for machine learning*. Morgan Kaufmann.

Walonoski, J.A., Heffernan, N.T. (2006) Detection and Analysis of Off-Task Gaming Behavior in Intelligent Tutoring Systems. In: *Proc. 8th Int. Conf. Intelligent Tutoring Systems* (pp. 382-391).