Labeling Student Behavior Faster and More Precisely with Text Replays

Ryan S.J.d. Baker[1], Adriana M.J.A. de Carvalho
rsbaker@cmu.edu, dikajoazeirodebaker@gmail.com
[1] Human Computer Interaction Institute, Carnegie Mellon University

Abstract. We present text replays, a method for generating labels that can be used to train classifiers of student behavior. We use this method to label data as to whether students are gaming the system, within 20 intelligent tutor units on Algebra. Text replays are 2-6 times faster per label than previous methods for generating labels, such as quantitative field observations and screen replays; however, being able to generate classifiers on retrospective data at the coder's convenience (rather than being dependent on visits to schools) makes this method about 40 times faster than quantitative field observations. Text replays also give precise predictions of student behavior at multiple grain-sizes, allowing the use of both hierarchical classifiers such as Latent Response Models (LRMs), and non-hierarchical classifiers such as Decision Trees. Training using text replay data appears to lead to better classifiers: LRMs trained using text replay data achieve higher correlation and A' than LRMs trained using quantitative field observations; Decision Trees are more precise than LRMs at identifying exactly when the behavior occurs.

1 Introduction

In recent years, there has been increasing interest in classifying what type of behavior a student is engaging in as he or she uses interactive learning software. For instance, models that can classify whether a student is gaming the system (attempting to succeed in an interactive learning environment by exploiting properties of the system rather than by learning the material [cf. 5]) have been developed for several learning systems [1, 6, 9, 20]. Models that can classify other types of behavior, such as help avoidance [1] and deep and shallow exploration behaviors [2] have also been developed. These models have been used as inputs to data mining analyses of student learning and motivation [8, 12], and have been used to select pedagogical interventions [3, 4, 20].

However, developing classifiers of student behavior has thus far been either highly time-consuming or has resulted in classifiers that are difficult to validate. In this paper, we present a method for quickly and accurately labeling data in terms of student behavior – text replays. The entire process of creating a detector for a unit in an intelligent tutor is about 40 times faster when text replays are used, compared to prior data labeling methods such as quantitative field observations or screen replays. Text replays also produce data which can be more easily used with off-the-shelf classification algorithms, such as those in Weka [21], than field observations. We validate this method by using it to develop classifiers of gaming the system for 20 different Cognitive Tutor lessons on algebra, and study those detectors' ability to predict which students game, how much they game, and when they game.

2 Prior Work

Several approaches have previously been used to develop classifiers of student behaviors, such as gaming the system, in interactive learning software.

One approach to developing classifiers of student behaviors has been to conduct quantitative field observations [5,17] of student behavior in the classroom (or other naturalistic settings), and then use classification algorithms on the resultant data [cf. 6, 20]. However, there are two drawbacks to this approach: First, this method is highly time-consuming, as it requires coders to travel to the setting of use and watch student behavior in real-time. Baker et al. [5] report making 563 classifications in around 7.5 hours of class time. However, since the observations in Baker et al. were made in classrooms distant from the researchers' offices, driving time and setup time should also be counted as part of the overall time cost of conducting these observations. If we assume around a half hour driving time in each direction, and 15 minutes setup time before each session, that works out to around 6 hours logistical time. Hence, while each observation took 20 seconds to conduct, the actual time cost per classification is 1.3 minutes. Additionally, this measure does not account for the difficulty of scheduling classroom observations within schools, and time lost to waiting for observation opportunities. Second, the resulting data labels are not immediately synchronized with the log data. Whereas log data is generally organized either at the semantic level of student actions (i.e. entering an answer, requesting help, etc.) or keystrokes/mouse movements [cf. 11]), the observations occupy 20-30 second time windows. In many cases, it is difficult to exactly synchronize these two types of data, since even a slight skew in time recording mechanisms may significantly affect accuracy. To compensate, researchers have probabilistically assigned time windows [cf. 20] or used hierarchical models that fit to overall proportions of behavior rather than labeling individual actions [cf.6].

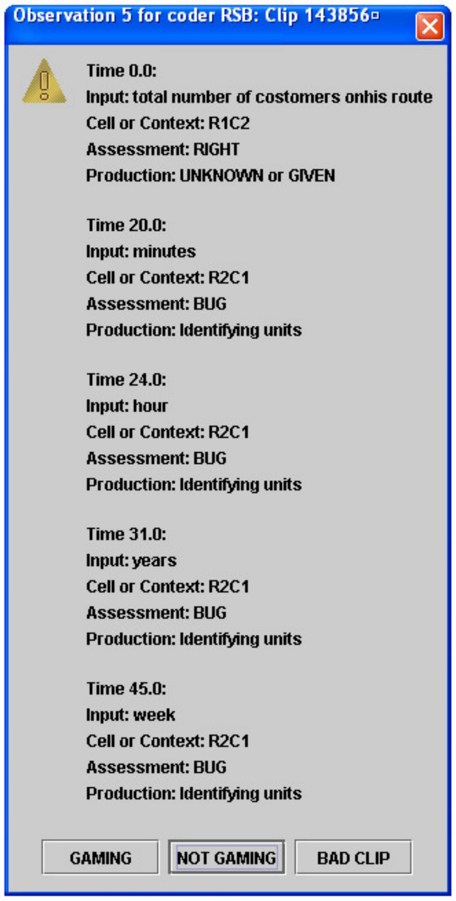
A second approach to developing classifiers of student behavior is to use knowledge engineering rather than machine learning on labeled data [cf.1,9]. This approach is generally quite quick to implement, but it is not possible to directly validate the accuracy of classifications without obtaining labeled data of some sort. The single example of a validation of a knowledge engineering behavior classifier in a learning environment (that we are aware of) involved the use of data collected through quantitative field observations [cf. 18]. A related approach is to use clustering to develop behavior categories, and then develop a classifier based on those behavior categories [cf. 2]. Like knowledge engineering, this approach is quick to implement, but suffers from the same difficulty in model validation.

An alternate approach is to directly label the log files with the construct(s) of interest. One method for doing so, proposed by de Vicente and Pain [11], is screen replays – labeling a student's behavior or affect while watching a video capture of the student's screen during learning software usage. This approach avoids synchronization problems in labeling data, as well as scheduling/driving time, but is still quite time-consuming, as coders watch student behavior in real time. de Vicente and Pain [11] report observers making 85 classifications in around 6 hours, an average of one classification every 4.2

minutes. Cocea and Weibelzahl [10] also report having human coders label log files, although they do not report the coding process in full detail.

3 Text Replays

We propose another method for directly labeling log files with the constructs of interest: text replays. Text replays represent a segment of student behavior from the log files in a textual ("pretty-printed") form. A sequence of actions of a pre-selected duration (in terms of time or length) is shown in a textual format that gives information about the actions and their context. In the example shown in Figure 1, the coder sees each action's time (relative to the first action in the clip), the problem context, the input entered, the relevant skill (production), and how the system assessed the action (correct, incorrect, a help request, or a "bug"/ misconception). The coder can then choose one of a set of behavior categories (in this study, gaming or not gaming), or indicate that something has gone wrong, making the clip uncodeable.


Observation 5 for coder RSB: Clip 143856

Time 0.0:
Input: total number of customers onhis route
Cell or Context: R1C2
Assessment: RIGHT
Production: UNKNOWN or GIVEN

Time 20.0:
Input: minutes
Cell or Context: R2C1
Assessment: BUG
Production: Identifying units

Time 24.0:
Input: hour
Cell or Context: R2C1
Assessment: BUG
Production: Identifying units

Time 31.0:
Input: years
Cell or Context: R2C1
Assessment: BUG
Production: Identifying units

Time 45.0:
Input: week
Cell or Context: R2C1
Assessment: BUG
Production: Identifying units

Figure 1: A "text replay" of student gaming

Text replays give relatively limited information, compared to quantitative field observations or full screen replays, omitting the visual information that would be seen in those methods (including mouse movements and partial responses). It is also necessary for the coder to understand the user interface in order to interpret contextual information. However, text replays are likely to be very quick to classify, and can be generated automatically from existing log files.

Text replays were first proposed in Baker, Corbett, & Wagner [7], where it was shown that text replays have good inter-rater reliability, and that text replays agree well with predictions made by models generated using data from quantitative field observations. In this paper, we test this technique's speed for labeling large amounts of data, develop classifiers of student behavior, and test these classifiers' goodness of fit.

4 Text Replay Labeling Study

We used text replays to label student behavior within a Cognitive Tutor for Algebra. Cognitive Tutors are a popular type of interactive learning environment now used by around half a million students a year in the USA. In Cognitive Tutors, students solve problems, with exercises and feedback chosen based on a model of which skills the student possesses. Cognitive Tutor Algebra has been shown to significantly improve student performance on standardized exams and tests of problem-solving skill [15].

Within our text replays, we labeled whether a student was gaming the system. We chose gaming the system, because it is a behavior for which multiple classifiers have been developed and validated [6, 18, 20]. Within Cognitive Tutor Algebra, as with other Cognitive Tutors, gaming the system consists of systematic guessing and hint abuse (clicking through hints at high speed until the system gives the answer).

We obtained data from students' use of Cognitive Tutor Algebra from the Pittsburgh Science of Learning Center DataShop (https://learnlab.web.cmu.edu/datashop/). The DataShop is a public resource for the learning science community, giving free access to anonymized data sets of student use of learning software. We used the "Algebra I 2005-2006 (Hampton only)" data set, a data set consisting of 436,816 student actions (an action is defined as entering an answer or requesting help) made by 59 students over the course of an entire school year, in a high school in the suburbs of a city in the Northeastern USA. Data from 32 tutor lessons was obtained; however, 12 lessons were eliminated from consideration for one of two reasons: either having insufficient data to be able to conduct at least 498 observations (500 was the planned cut-off, but one lesson with 498 observations was included), or gaming the system never being coded in that lesson.

Across lessons, 18,737 text replays were labeled. These text replays were randomly selected; the chance of a specific sequence of actions being chosen was weighted, for each lesson, according to the total number of actions in that lesson. An average of 887 text replay observations were made in each lesson (SD=178; min=498; max=1075).

A single coder (the second author) labeled all 18,737 text replays. This coder was trained by the first author, who had previously used the technique in [7]. Training consisted of repeatedly coding the same clips, and comparing and discussing the results, until 100% agreement was obtained on two sets in a row. Text replays of gaming previously achieved reasonably high inter-reliability (Cohen's κ) of 0.58, in a study where coders did not engage in this training practice [7]. Training took approximately two hours.

The coder labeled the 18,737 text replays in 206 hours of work, a rate of about 39.6 seconds per observation. The 206 hours included time spent studying each tutor lesson's user interface. This was about twice as fast, label per label, as quantitative field observations of gaming conducted in prior research [7], and about 6 times as fast as observations of affect conducted using screen replays [11]. The observations were conducted across 4.5 months, at the coder's convenience, rather than during an entire school year during school hours. The time savings from conducting the observations when convenient for the coder – rather than when convenient for the school – was quite substantial. 4.5 months of work was able to generate data to build gaming classifiers for 20 tutor lessons (the classifiers will be discussed in the next section). By comparison, 3 years of work using quantitative field observations to label student data only produced enough data to develop classifiers of gaming for 4 tutor lessons [6]. Hence, text replays enabled us to collect data for 5 times as many classifiers in 12.5% as much total time – around 40 times faster. This result suggests that the text replay method is substantially more efficient than prior methods for generating labels of gaming the system, beyond just the time spent on each individual label.

5 Developing Classifiers of Gaming the System

Once the text replays were obtained, we developed two types of classifiers of gaming the system. First, we built classifiers of gaming using a Latent Response Model(LRM) [16]. In this approach, discussed in considerably more detail in [6], the model predicts whether each student action is gaming, then aggregates those predictions to predict each student's frequency of gaming (and trains on the overall proportion of time each student spent gaming, rather than training on labels of individual actions). Training detectors in this fashion enables us to study whether detectors developed using data from text replays are as accurate as detectors previously developed using data from quantitative field observations [6], using the same classification method.

Second, we took advantage of the fact that text replays label individual actions in the logs as gaming or not gaming, to use a second method. We built gaming detectors using J48 decision trees, the same technique used in Walonoski & Heffernan [20], using the Weka data mining package [21]. This method was slightly to moderately more accurate at classifying gaming the system than other classification algorithms within the Weka package. It is worth noting that these labels of individual actions cannot be considered perfectly accurate, since the observer labeled a clip as "gaming" if any of the actions in the clip involved gaming. Therefore, actions at the beginning or end of clips may not in all cases be instances of gaming. This suggests that, within this data set, a 100% perfect match between our classifier's labels of individual actions and those actions' labels (which will be measured with Kappa) is not necessary (or desirable). This limitation could be addressed in future work by having observers explicitly label which actions in a clip are gaming, but would have the cost of reducing the method's speed.

Within each approach, a separate classifier was trained for each lesson and tested in that lesson. Due to the large number of models fit, the training set was used as the test set in all cases, for tractability. As this testing method risks some over-fitting, pruning was used on the J48 trees, and model size was capped for the latent response models.

26 features were distilled for each action in the log files for use by the classification algorithms, including features involving details about the action (Was it correct or incorrect, or a help request? Was it the first attempt at the step?), assessments of student knowledge, information about the time the student took on this step (absolute, and relative to other students), and information about the student's previous interaction behaviors (How often has this student gotten this step wrong in the past? How much help has the student requested?). A full list of features can be found in [6].

6 Classifier Goodness

We assessed the two types of classifiers, for each lesson, using three metrics: A', correlation, and kappa. A' addresses a classifier's ability to assess which students gamed. A' is the probability that if the detector is comparing two students, one who gamed and one who never gamed, it will correctly identify which student is which. A' is equivalent to both the area under the ROC curve in signal detection theory, and to W, the Wilcoxon statistic [14]. A model with an A' of 0.5 performs at chance, and a model with an A' of

Table 1. The performance of each model on each metric, across models (standard errors given in parentheses). Both models are statistically better than chance on all metrics; boldface indicates a model is statistically significantly better than the other model, for that metric.

	A'	Correlation	Kappa
Latent Response Model	**0.96 (0.02)**	**0.90 (0.02)**	0.04 (0.01)
Decision Tree	0.69 (0.03)	0.44 (0.05)	**0.40 (0.05)**

1.0 performs perfectly. Correlation assesses whether the detector predicts the correct amount of gaming for each student. Kappa assesses whether the detector accurately identifies the correct actions as gaming. A Kappa of 0 indicates that the detector performs at chance, and a Kappa of 1 indicates that the detector performs perfectly. The full pattern of results is shown in Table 1.

For A', the Latent Response Models averaged an A' of 0.96 (SE=0.02) across the 20 lessons, meaning that they could distinguish a gaming student from a non-gaming student in the data 96% of the time. This A' is considerably higher than the A' of previous models of gaming the system developed using data from quantitative field observations (in that earlier work, models averaged an A' of 0.86 on the training set) [6]. The Decision Trees averaged an A' of 0.69 (SE=0.03). We can calculate the statistical significance of the difference of these models from chance (and from each other) by computing the standard error of each A' estimate [14], conducting Z tests, and then using Strube's Adjusted Z to aggregate across lessons in a fashion that controls for within-student non-independence [20, Appendix II in 6]. We find that the LRM A' is significantly higher than chance, Z=26.56, two tailed $p<0.0001$, that the Decision Tree A' is also significantly higher than chance, Z=3.24, two tailed $p<0.01$, and that the LRM A' is significantly higher than the Decision Tree A' across lessons, Z=3.35, two tailed $p<0.01$.

For correlation, the Latent Response Models averaged a correlation of 0.90 (SE=0.02) across the 20 lessons, between the predicted and actual frequencies of gaming, and the Decision Trees averaged a correlation of 0.44 (SE=0.05). Statistical significance can be computed by computing the significance of each correlation (or the significance of differences in correlation for correlated samples – [cf. 13]), converting those significance values to Z scores, and then aggregating across Z scores using Strube's Adjusted Z. We find that the LRM correlation is significantly higher than chance, Z=10.04, two tailed $p<0.0001$, that the Decision Tree correlation is also significantly higher than chance, Z=3.46,two-tailed $p<0.0001$, and that the LRM correlation is significantly higher than the Decision Tree correlation across lessons, Z=5.68, two tailed $p<0.0001$.

For kappa, the Latent Response Models averaged a kappa of 0.04 (SE=0.01) across the 20 lessons in predicting the labels of whether individual actions involved gaming, and the Decision Trees averaged a kappa of 0.40 (SE=0.05). The statistical test most closely corresponding to kappa is chi-squared; however, this data violates the independence assumptions of chi-squared. Thus, we instead do t-tests on the kappa values for each lesson (either one-sample t-tests or paired t-tests), convert the significance values to Z scores, and then aggregating across Z scores using Strube's Adjusted Z. This method treats the kappa values as point-estimates rather than as having a standard error, effectively giving us a sample size of 20 (the number of lessons) rather than the full sample size – in other words, this test is overly conservative, and should err in the direction of non-significance. Despite the poor kappa for LRM, it is significantly higher than chance, $t(19)=3.77$, two tailed $p<0.01$. The Decision Tree kappa is also significantly higher than chance, $t(19)=7.81$,two-tailed $p<0.0001$. The Decision Tree kappa is significantly higher than the LRM kappa across lessons, $t(19)=7.45$, two tailed $p<0.0001$.

The poor performance of the LRM-based classifiers on kappa is surprising, given the high performance on the other metrics, and the effectiveness of this technique at assigning interventions to students which improve learning [4]. One potential explanation is that the model's use of historical features leads it to identify gaming in the correct skills, but on transactions after the transaction where gaming actually occurred (i.e. the next time the skill is encountered), a pattern observed in previous classifiers of gaming trained in this fashion [6]. Indeed, we find that the LRM-based classifier is significantly more likely to detect gaming on skills the student has previously gamed (i.e. detecting gaming after the fact), than on other skills, $t(58) = -4.77$, two-tailed $p<0.001$, for a paired t-test. Classifiers with this characteristic will be useful for some types of interventions, such as giving supplementary exercises on material a student bypassed by gaming [4] or giving metacognitive interventions to gaming students between problems [3], but will be less appropriate for more immediate types of interventions. Interventions which depend on occurring exactly after a student games should probably be assigned by detectors trained on individual actions rather than overall proportions of behavior. On the other hand, the LRM-based classifier's higher ability to identify gaming students is also relevant, since it increases the probability of assigning interventions to the students who most need them. Similarly, the degree to which this limitation will affect this classifier's usefulness for data mining analyses of motivation or learning depends on the analysis. Analyses at broader grain-sizes which look at the overall incidence of gaming [e.g. 8, 12] may benefit from using an LRM-based classifier, whereas analyses that depend on accurate assessments of whether each action involves gaming may benefit more from using a Decision Tree.

One potential solution that could avoid the limitations of the classifiers presented here would be to develop a hierarchical classifier (such as LRM) which optimizes on measures at both grain-sizes during training (e.g. using all three measures presented here). Such a classifier could potentially succeed both at detecting when an individual student is gaming, and distinguishing with maximum accuracy which students game and how much they game. However, this classifier would require the type of labels at multiple grain-sizes that Text Replays can provide.

7 Discussion and Conclusions

In this paper, we have presented text replays, a method for labeling student's behavior within log files. When all factors are taken into account, including the time needed to make individual labels, and the logistics involved in conducting studies in schools, text replays appear to speed the process of data collection by around 40 times compared to quantitative field observations, and at least 6 times compared to screen replays.

Where this technique applies, therefore, it may make classifier development accessible to a much larger pool of researchers and developers. However, this technique may not be applicable for coding every type of construct that can be coded with quantitative field observations or screen replays. Those techniques have been used to code students' affect and motivation [cf. 11, 17]; it is likely that some affective and motivational constructs will be difficult to code using text replays. Establishing which categories of behavior can accurately be labeled in text replays will be an important area for future work.

Unlike quantitative field observations, text replays make it easy to label individual actions as gaming or not gaming, though these labels are not 100% accurate, as clips often do not coincide exactly with the beginning and end of a gaming episode. Nonetheless, these action-by-action labels make it much more feasible to use off-the-shelf classifiers such as J48 Decision Trees to train gaming detectors which are reasonably successful at identifying exactly when students game, which students are gaming, and how much each student is gaming. Being able to use off-the-shelf classification algorithms available in Weka should make it more feasible for researchers and developers without machine learning experience to develop classifiers which are useful for learning analyses and to drive interventions by learning software.

An alternate modeling framework, Latent Response Models, was trained on each student's frequency of gaming. LRMs, as implemented here, were much more successful at identifying which students gamed, and how much each student gamed, but much less successful at identifying exactly when students game. Interestingly, the LRMs developed using text replay data appeared to be more accurate than prior LRMs developed using quantitative field observations. One explanation is that the text replay method both enables more precise assessments of students' gaming frequency (via making it easy to collect more labels), and allows a coder to take extra time on more ambiguous cases (while working quickly on clear cases).

The pattern of successes of the classifiers trained using different data suggests that it may be necessary to develop an algorithm that uses both overall gaming frequency data and action-by-action labels of gaming during the training processes in order to optimally capture both which students game, and when they game.

One curious aspect of the models developed here is that they do not replicate the split between types of gaming found in [6]. In that work, LRMs trained to classify gaming students systematically only captured a proportion of students – the students who gamed and had poorer learning. (A separate classifier was trained to detect students who gamed but did not have poorer learning). There was no evidence for such a split in this study, where LRM classifiers had average A' values of 0.96. It is not yet clear whether the split in gaming behavior in [6] was idiosyncratic to the learning system studied, or whether this effect is dependent on differences in the labeling method in some subtle fashion.

In conclusion, text replays appear to considerably speed the process of collecting the data necessary to develop effective classifiers of student behavior. Making it easier to develop classifiers of student behavior will expand the benefits of this technique to a wider pool of researchers, facilitating the development of more adaptive educational software and the use of data mining techniques to conduct complex analyses on student learning.

Acknowledgements

We would like to thank Mercedes Rodrigo and Albert Corbett for helpful suggestions and discussions. This work was funded by the Pittsburgh Science of Learning Center, National Science Foundation award SBE-0354420.

References

[1] Aleven, V., McLaren, B.M., Roll, I., and Koedinger, K.R. Toward tutoring help seeking: Applying cognitive modeling to meta-cognitive skills. *Proceedings of the 7th International Conference on Intelligent Tutoring Systems (ITS 2004)*, 227-239.

[2] Amershi, S., Conati, C. Automatic Recognition of Learner Groups in Exploratory Learning Environments. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS 2006)*, 463-472.

[3] Arroyo, I., Ferguson, K., Johns, J., Dragon, T., Meheranian, H., Fisher, D., Barto, A., Mahadevan, S., and Woolf. B.P. Repairing Disengagement with Non-Invasive Interventions. *Proceedings of the 13h International Conference on Artificial Intelligence in Education (AIED-2007)*, 195-202.

[4] Baker, R.S.J.d., Corbett, A.T., Koedinger, K.R., Evenson, E., Roll, I., Wagner, A.Z., Naim, M., Raspat, J., Baker, D.J., Beck, J. Adapting to When Students Game an Intelligent Tutoring System. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS 2006)*, 392-401.

[5] Baker, R.S., Corbett, A.T., Koedinger, K.R., Wagner, A.Z. Off-Task Behavior in the Cognitive Tutor Classroom: When Students "Game The System". *Proceedings of ACM CHI 2004: Computer-Human Interaction*, 383-390.

[6] Baker, R.S.J.d., Corbett, A.T., Roll, I., Koedinger, K.R. (to appear) *Developing a Generalizable Detector of When Students Game the System* To appear in User Modeling and User-Adapted Interaction. Online at http://www.cs.cmu.edu/~rsbaker/USER475.pdf

[7] Baker, R.S.J.d., Corbett, A.T., Wagner, A.Z. Human Classification of Low-Fidelity Replays of Student Actions. *Proceedings of the Educational Data Mining Workshop at the 8th International Conference on Intelligent Tutoring Systems*, 2006, 29-36.

[8] Baker, R., Walonoski, J., Heffernan, N., Roll, I., Corbett, A., Koedinger, K. *Why Students Engage in "Gaming the System" Behavior in Interactive Learning Environments*. Journal of Interactive Learning Research, 2008, 19 (2), 185-224.

[9] Beck, J. Engagement tracing: using response times to model student disengagement. *Proceedings of the 12th International Conference on Artificial Intelligence in Education (AIED 2005)*, Amsterdam, pp. 88-95.

[10] Cocea, M., Weibelzahl, S. Eliciting Motivation Knowledge from Log Files Towards Motivation Diagnosis for Adaptive Systems. *Proceedings of 11th International Conference on User Modeling, UM2007,* 197-296.

[11] de Vicente, A. and Pain, H. Informing the detection of the students' motivational state: an empirical study. *Proceedings of the Sixth International Conference on Intelligent Tutoring Systems*, 2002, 933-943.

[12] Feng, M., Heffernan, N.T., Koedinger, K.R. Looking for Sources of Error in Predicting Student's Knowledge. *Educational Data Mining: Papers from the 2005 AAAI Workshop,* 54-61.

[13] Ferguson, G.A. *Statistical Analysis in Psychology and Education,* 1971. New York: McGraw-Hill.

[14] Hanley, J.A., McNeil, B.J. The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology,* 1982, 143, 29-36.

[15] Koedinger, K. R., & Corbett, A. T. Cognitive tutors: Technology bringing learning sciences to the classroom. In R. K. Sawyer (Ed.), *The Cambridge handbook of the learning sciences*, 2006, pp. 61-77. New York, NY: Cambridge University Press.

[16] Maris, E. Psychometric Latent Response Models. *Psychometrika*, 1995, 60 (4), 523-547.

[17] Rodrigo, M.M.T., Baker, R.S.J.d., Lagud, M.C.V., Lim, S.A.L., Macapanpan, A.F., Pascua, S.A.M.S., Santillano, J.Q., Sevilla, L.R.S., Sugay, J.O., Tep, S., Viehland, N.J.B. Affect and Usage Choices in Simulation Problem Solving Environments. *Proceedings of Artificial Intelligence in Education 2007*, 145-152.

[18] Roll, R., Baker, R., Aleven, V., McLaren, B., Koedinger, K. Modeling Students' Metacognitive Errors in Two Intelligent Tutoring Systems. *Proceedings of User Modeling 2005*, 367-376.

[19] Strube, M.J. Combining and comparing significance levels from nonindependent hypothesis tests. *Psychological Bulletin*, 1985, 97, 334-341.

[20] Walonoski, J.A., Heffernan, N.T. Detection and Analysis of Off-Task Gaming Behavior in Intelligent Tutoring Systems. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems,* 2006, 382-391.

[21] Witten, I.H., Frank, E. *Data Mining: Practical machine learning tools and techniques,* 2005. San Francisco: Morgan Kaufmann.