User Model User-Adap Inter
DOI 10.1007/s11257-007-9045-6

ORIGINAL PAPER

Developing a generalizable detector of when students game the system

Ryan S. J. d. Baker · Albert T. Corbett · Ido Roll · Kenneth R. Koedinger

Received: 12 August 2006 / Accepted in revised form: 11 November 2007

Abstract Some students, when working in interactive learning environments, attempt to "game the system", attempting to succeed in the environment by exploiting properties of the system rather than by learning the material and trying to use that knowledge to answer correctly. In this paper, we present a system that can accurately detect whether a student is gaming the system, within a Cognitive Tutor mathematics curricula. Our detector also distinguishes between two distinct types of gaming which are associated with different learning outcomes. We explore this detector's generalizability, and find that it transfers successfully to both new students and new tutor lessons.

Keywords Gaming the system · Latent response models · Cognitive tutors · Behavior detection · Machine learning · Generalizable models · Student modeling · Interactive learning environments

1 Introduction

Developing systems that can reliably identify differences in how students choose to use interactive learning environments, and the attitudes and goals which underlie these

R. S. J. d. Baker (✉) · A. T. Corbett · I. Roll · K. R. Koedinger
Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
e-mail: rsbaker@cmu.edu

A. T. Corbett
e-mail: corbett@cmu.edu

I. Roll
e-mail: idoroll@cmu.edu

K. R. Koedinger
e-mail: koedinger@cmu.edu

decisions, is an interesting and challenging problem which has received considerable attention in recent years (Aleven et al. 2004; Arroyo and Woolf 2005; Beck 2005; Conati and McLaren 2005; D'Mello et al. in press; de Vicente and Pain 2002; Johns and Woolf 2006; Walonoski and Heffernan 2006a).

One behavior that has been the subject of particular interest in recent years is "gaming the system", defined as "attempting to succeed in an educational environment by exploiting properties of the system rather than by learning the material and trying to use that knowledge to answer correctly" (Baker et al. 2006). Gaming behaviors have been observed in a variety of types of learning environments, from educational games (Magnussen and Misfeldt 2004) to online course discussion forums (Cheng and Vassileva 2005). Though gaming behavior had been documented in computer-assisted instruction as early as the early 1970s (Tait et al. 1973) and again in the 1990s (Schofield 1995; Wood and Wood 1999), the topic has seen a burst of attention in the last four years within the context of intelligent tutoring systems (cf. Schofield 1995; Wood and Wood 1999; Aleven 2001; Mostow et al. 2002; Baker et al. 2004; Beck 2005; Murray and vanLehn 2005; Beal et al. 2006; Johns and Woolf 2006; Walonoski and Heffernan 2006a), after it was demonstrated that gaming behavior is associated with significantly poorer learning in Cognitive Tutor classes (Baker et al. 2004) and after the first systems that could accurately detect gaming behavior were reported, Baker et al.'s (2004) Gaming Detector, and Aleven et al.'s (2004) Help-Seeking Tutor Agent. In the three years since those two systems were simultaneously reported at Intelligent Tutoring Systems 2004, at least four other independently developed systems which detect gaming behavior have been reported at scientific conferences (Beck 2005; Walonoski and Heffernan 2006a; Johns and Woolf 2006; Beal et al. 2006).

In this paper, we will discuss one of the first two gaming detectors developed, Baker et al.'s (2004) Gaming Detector, presenting it in its most current form. We will discuss the data used to develop the Gaming Detector, and evidence that the Gaming Detector can effectively detect gaming, distinguish between types of gaming, and generalize to new tutor lessons. We will conclude with a comparison between our research group's Gaming Detector, and systems that detect gaming behavior which were developed by other research groups.

2 Gaming the system in cognitive tutors

In this paper, we will discuss work on detecting gaming within Cognitive Tutors. Cognitive Tutor learning environments are designed to promote learning by doing. Within the Cognitive Tutor environments discussed within this paper, each student individually completes mathematics problems. The Cognitive Tutor environment breaks down each mathematics problem into the steps of the process used to solve the problem, making the student's thinking visible. As a student works through a problem, a running cognitive model assesses whether the student's answers map to correct understanding or to a known misconception (cf. Anderson et al. 1995). If the student's answer is incorrect, the answer turns red; if the student's answers are indicative of a known misconception, the student is given a "buggy message" indicating how their current knowledge differs from correct understanding. Cognitive Tutors

also have multi-step hint features; a student who is struggling can ask for a hint. He or she first receives a conceptual hint, and can then request further hints, which become more and more specific until the student is given the answer (see Fig. 1). The hints are context-sensitive and tailored to the exact problem step the student is working on. As the student works through the problems in a specific curricular area, the system uses Bayesian knowledge-tracing (Corbett and Anderson 1995) to determine which skills that student is having difficulty with, calculating the probability that the student knows each skill based on that student's history of responses within the tutor. Using these estimates of student knowledge, the tutoring system gives each student problems which are relevant to the skills which he or she is having difficulty with.

Cognitive Tutor material is typically structured into independent lessons, each of which covers a set of related skills and concepts. Year-long courses are composed of sequences of lessons, where the knowledge in later lessons generally builds upon the knowledge in previous lessons. Year-long Cognitive Tutor courses were used in over 1,000 U.S. high schools as of the 2005–2006 school year, for a variety of mathematical subjects such as Algebra, Pre-Algebra, and Geometry.

Within Cognitive Tutors, gaming the system consists of the following behaviors:

1. quickly and repeatedly asking for help until the tutor gives the student the correct answer (cf. Aleven 2001)
2. inputting answers quickly and systematically. For instance, systematically guessing numbers in order (1,2,3,4…) or clicking every checkbox within a set of multiple-choice answers, until the tutor identifies a correct answer and allows the student to advance.

These categories of behavior appear to be common to other intelligent tutoring systems as well (Beck 2005; Murray and vanLehn 2005; Walonoski and Heffernan 2006a; Johns and Woolf 2006). Other examples of gaming the system include choosing to work on material which the student has already memorized (Mostow et al. 2002), and intentionally posting irrelevant material to online course discussion forums where participation is automatically graded (Cheng and Vassileva 2005).

In our early work to develop a gaming detector (Baker et al. 2004), using only a single tutor lesson from a middle school mathematics curriculum (cf. Koedinger 2002), we serendipitously found evidence suggesting that gaming divides into two distinct categories of behavior. A detector trained to detect all gaming students only succeeded in detecting gaming in about half of the students observed gaming in the study. Further investigation produced evidence that the detector was only detecting students who both gamed and had low post-test scores. The detector was not detecting the students who gamed but had high post-test scores (regardless of whether they had low pre-test scores, suggesting that they learned from the tutor, or high pre-test scores, suggesting that they already knew the material) (cf. Baker et al. 2004).

Follow-up analysis, using a broader data set, produced further evidence suggesting that students who gamed and had low post-test scores differ considerably from students who gamed and had high post-test scores. A detector trained on either of the two groups of gaming students accurately captured that group (using leave-out-one-cross-validation, where a detector is trained on every student except one, and then used to

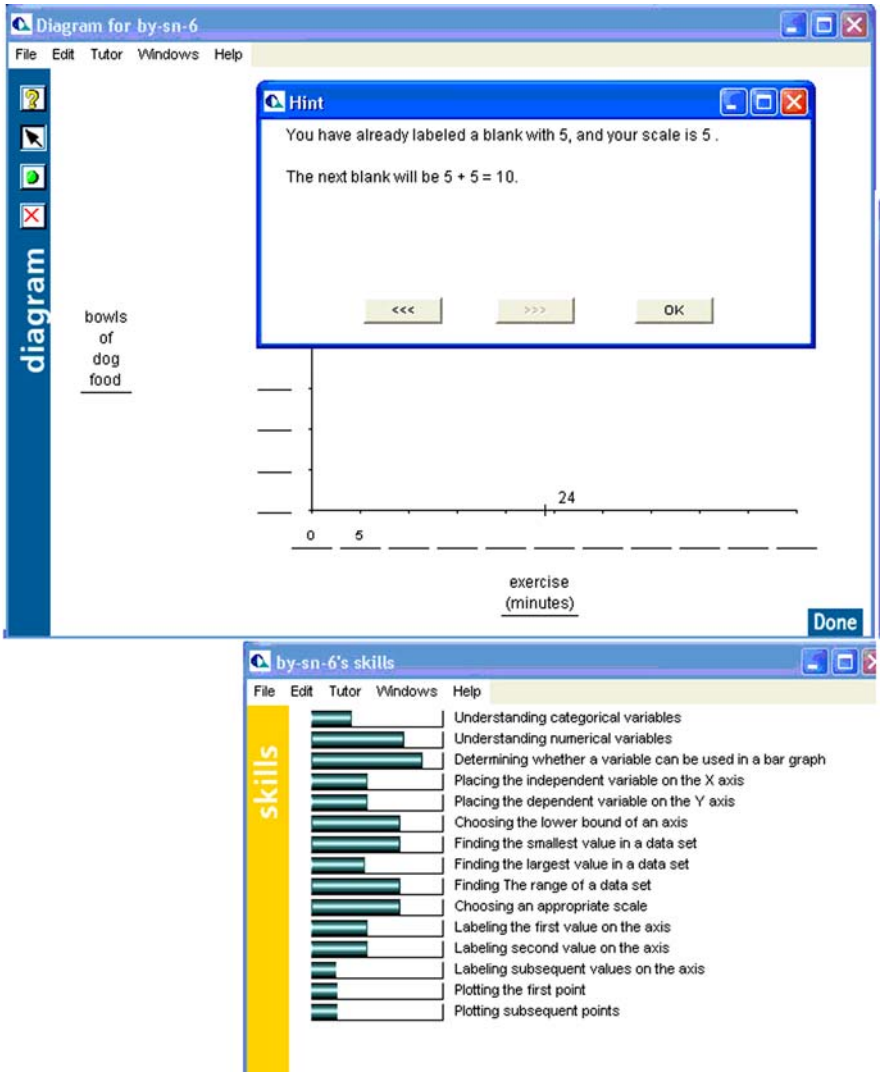

Fig. 1 The last stage of a multi-stage hint in the tutor lesson on scatterplots: The student labels the graph's axes and plots points in the upper window; the tutor's estimates of the student's skills are shown in the lower window; the hint window (superimposed on the upper window) allows the tutor to give the student feedback

make a prediction about the student left-out), but generally did not capture students from the other gaming group, suggesting that these two categories of behavior are indeed differentiable within students using Cognitive Tutors. We will present some of this evidence later in the paper.

In this paper we focus on the detector that captures the behavior of the group of students who gamed the system in the fashion associated with poorer learning, as

Table 1 Data obtained for each tutor lesson

Lesson	Number students	Number	Pct. of students	
			GAMED-HURT (%)	GAMED-NOT-HURT (%)
Scatterplot	237	71,232	8	25
Probability	50	15,858	10	6
Geometry	111	30,991	27	3
Percents	38	10,135	8	5

this "harmful" type of gaming[1] is associated with a concrete difference in learning outcomes while other gaming behavior does not appear to be. This does not imply that it is not important to understand why some gaming behaviors are not associated with lower learning gains. However, developing an accurate and generalizable detector of harmful gaming behavior has considerably more potential to increase intelligent tutors' educational effectiveness (since such a detector can be used to respond to such behavior and potentially help harmfully gaming students learn better) than the development of an accurate and generalizable detector of other types of gaming behavior does.

3 Data

The first detector of gaming (presented in Baker et al. 2004) was developed using data from a intelligent tutor lesson on scatterplots, drawn from a Cognitive Tutor curriculum on middle school mathematics (Koedinger 2002).

In order to study issues of generalizability in gaming detection, we collected data from three additional lessons from the same tutoring curriculum, lessons in the domains of geometry, percents, and probability, giving us data from four tutor lessons in total. All data came from classrooms (4–6 classrooms per year) in two school districts in suburban Pittsburgh.

The scatterplot lesson data was drawn from classes in 2003, 2004, and 2005. The geometry and probability lesson data was drawn from classes in 2004. The data for the geometry, probability, and scatterplot lesson (2004 cohort) involved the same group of students, with some non-overlap due to absence. The data for the percents lesson was drawn from classes in 2005, but in a fashion that did not result in any overlap between the students in the scatterplot and percents data. In total, data was collected from 436 student/lesson pairs. Each student completed an average of 297 actions in the tutor (SD = 132) per lesson, with extremes at 35 and 752 actions, for a total of 129,341 actions. The number of students and number of tutor actions logged for each lesson is given in Table 1.

For each of these lessons, we had the following data:

– Quantitative field observations, in order to estimate what percentage of time each student gamed the system. In quantitative field observations, one or more observers

[1] The word "harmful" is used for brevity and simplicity in discussion; there is still not conclusive evidence as to whether the relationship between harmful gaming and learning is causal or correlational.

make repeated 20-s observations of the behavior of a set of students as the students use tutoring software, coding each student's behavior during an observation according to a pre-determined set of categories (in this case, including gaming the system, off-task behavior, talking on-task, and working in the tutor). Observations were conducted using peripheral vision, and achieved acceptable inter-rater reliability ($\kappa = 0.74$). Full detail on the method used is given in (Baker et al. 2004).
– Pre-tests and post-tests for each lesson, to determine how much each student learned while using the tutor—in all cases, test items were counterbalanced across the pre-test and post-test.
– Detailed log-files of the students' interactions with the tutor (see below).

Data on learning gains was used to distinguish between the two types of gaming behavior, both during training and when evaluating goodness-of-fit. Specifically, among the students observed gaming, a student was labeled GAMED-NOT-HURT (i.e., gaming in the non-harmful fashion) if he or she had a sizeable pre-post gain (at least two more skills demonstrated correctly on the post-test than on the pre-test), or a pre-test score high enough to make it impossible to gain two skills (i.e. perfect or only one error). Gaming students who had low scores on both the pre-test and post-test were labeled GAMED-HURT (i.e., gaming in the fashion associated with poorer learning).

Log files of each student's actions within the tutor were used in order to develop a model relating specific features of student actions to the overall construct of harmful gaming. For each student action recorded in the log files, a set of 26 features describing that student action were distilled. These features consisted of

- Details about the action
  - The tutoring software's assessment of the action—was the action correct, incorrect and indicating a known bug (procedural misconception), incorrect but not indicating a known bug, or a help request?
  - The type of interface widget involved in the action—was the student choosing from a pull-down menu, typing in a string, typing in a number, plotting a point, or selecting a checkbox?
  - Was this the student's first attempt to answer (or obtain help) on this problem step?
- Knowledge assessment
  - The tutor's assessment, after the action, of the probability that the student knows the skill involved in this action, called "pknow", derived using the Bayesian knowledge tracing algorithm in (Corbett and Anderson 1995).
  - "Pknow-direct", a fairly complicated feature found in its final form in the tutor log files. If the current action is the student's first attempt on this problem step, then pknow-direct is equal to pknow, but if the student has already made an attempt on this problem step, then pknow-direct is -1. Pknow-direct allows a contrast between a student's first attempt on a skill he/she knows very well and a student's later attempts.
  - Whether the action involved a skill which students, on the whole, knew before starting the tutor lesson, or failed to learn during the tutor lesson. (two variables)
- Time
  - How many seconds the action took.

  - The time taken for the action, expressed in terms of the number of standard deviations this action's time was faster or slower than the mean time taken by all students on this problem step, across problems.
  - The time taken in the last 3, or 5, actions, expressed as the sum of the numbers of standard deviations each action's time was faster or slower than the mean time taken by all students on that problem step, across problems. (two variables)
  - How many seconds the student spent on each opportunity to practice the primary skill involved in this action, averaged across problems.
- Previous interaction
  - The total number of times the student has gotten this specific problem step wrong, across all problems. (includes multiple attempts within one problem)
  - What percentage of past problems the student made errors on this problem step in
  - The number of times the student asked for help or made errors at this skill, including previous problems.
  - How many of the last 5 actions involved this problem step.
  - How many times the student asked for help in the last 8 actions.
  - How many errors the student made in the last 5 actions.

Due to logging errors, the log data from 2003 and 2004 lacked information on internal steps of hint requests. Features on internal steps of hint requests were distilled for the 2005 data, but did not significantly improve fit, and are not included in the analyses presented here.

4 The gaming detector

4.1 Detector structure

Latent Response Models (Maris 1995) were used as the statistical basis for the detector of harmful gaming. Latent Response Models have the advantage of easily and naturally integrating multiple data sources, at different grain sizes, into a single model.

A detector of gaming, in the framework used here, has one observable level and two hidden ("latent") levels. The model's overall structure is shown in Fig. 2. In a gaming detector's outermost/observable layer, the gaming detector assesses how frequently each of n students is gaming the system; those assessments are labeled $G'_0 \cdots G'_n$. The gaming detector's assessments for each student can then be compared to the observed proportions of time each student spent gaming the system, $G_0 \cdots G_n$ (the metrics used will be discussed within the model selection section). In order to develop a detector which only detects harmful gaming (as opposed to attempting to detect both types of gaming within a single detector), students labeled GAMED-NOT-HURT (because of high pre-post gain or a high pre-test score) were assigned a value of 0 for their proportion of time spent gaming harmfully.

The proportion of time each student spends gaming is assessed as follows: First, the detector makes a (binary) assessment as to whether each individual student action (denoted P'_m) is an instance of gaming. From these assessments, $G'_0 \cdots G'_n$ are derived by taking the percentage of actions which are assessed to be instances of gaming, for each student.

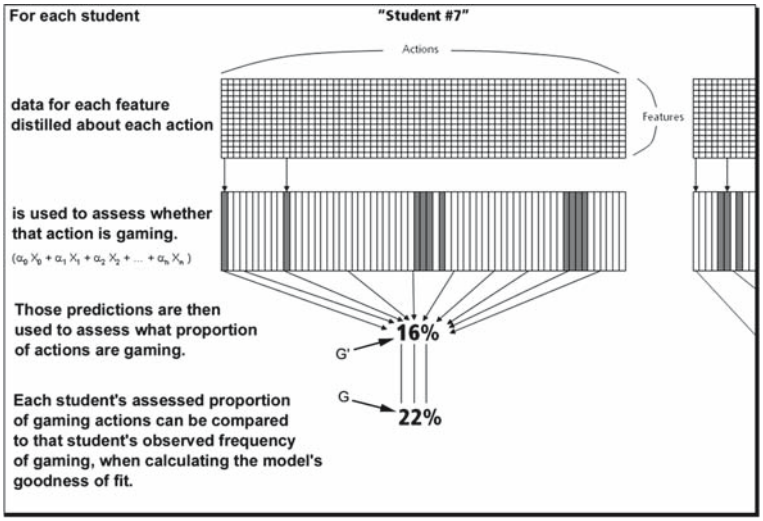

Fig. 2 The architecture of the gaming detector

An action is assessed to be gaming or not, by a function on parameters composed of the features drawn from each action's characteristics. Each parameter in a candidate model of gaming is either a linear effect on one feature (a parameter value α_i multiplied by the corresponding feature value X_i: $\alpha_i\ X_i$), a quadratic effect on one feature (parameter value α_i multiplied by feature value X_i, squared : $\alpha_i X_i^2$), or an interaction effect on two features (parameter value α_i multiplied by feature value X_i, multiplied by feature value X_j : $\alpha_i X_i X_j$).

An assessment H_m as to whether action m is an instance of gaming is computed as $H_m = \alpha_0\ X_0 + \alpha_1\ X_1 + \alpha_2\ X_2 + \cdots + \alpha_n\ X_n$, where α_i is a parameter value and X_i is the data value for the corresponding parameter (a single feature, that feature squared, or two features multiplied by each other), for this action, in the log files. The value given by the linear combination is the first hidden level and top layer in Fig. 2. Each assessment H_m is then thresholded using a step function, such that if $H_m \leq 0.5$, $H'_m = 0$, otherwise $H'_m = 1$. The set of thresholded values makes up the second hidden level and middle layer in Fig. 2. This gives us a set of classifications H'_m for each action within the tutor, which are then used to create the assessments of each student's proportion of gaming, $G'_0 \cdots G'_n$. These assessments of each student's proportion of gaming, which make up the observable level of the model (the bottom layer in Fig. 2), are compared to the observed values of gaming during model fitting and validation.

4.2 Detector selection

There is a very large space of potential models describing gaming behavior (if any model with 1–7 parameters is permitted, approximately 10^{13} models are possible—

this number is computed by taking the number of potential parameters and computing the number of combinations of 1–7 parameters).

A combination of Fast Correlation-Based Filtering (Yu and Liu 2003)[2] and Forward Selection (Ramsey and Schafer 1997) was used in order to efficiently search this space of models, as follows:

First, we conducted Fast Correlation-Based Filtering. The full set of possible single-parameter detectors was selected, using Iterative Gradient Descent (Boyd and Vandenberghe 2004) to find the best value for each parameter. From the full set of possible single-parameters, we selected a subset that fit the following two criteria:

1. Each single-parameter gaming detector was at least 60% as good as the best single-parameter detector found (in terms of linear correlation to the observed data).
2. If two parameters had a closer correlation than 0.7 to each other, only the better-fitting single-parameter detector was used.

Using Fast-Correlation Based Filtering enabled us to cut down search time considerably, since we were able to search a limited sub-set of the space while having reasonably high confidence that we were covering a representative sample of the entire space.

Once a set of single-parameter detectors was obtained, we expanded each detector, using Forward Selection. Using a simple Forward Selection procedure enabled us to search each path selected (which were together a representative sample of the entire space) in an exhaustive fashion. To each model, we tried adding each potential additional parameter, and selected the parameter that most improved the linear correlation between the detector's assessments and the original data. The process was repeated on each candidate detector until each candidate detector had seven parameters. In early work, Leave-One-Out-Cross-Validation (LOOCV) was used to determine the optimal model size, but this method became intractable when training many detectors with data from hundreds of students. We chose seven as a maximum model size, because many models had 4–7 parameters when cross validation was used, but 8 or more parameters were quite rare. Pseudo-code of the algorithm used for model selection is given in Appendix I.

This process resulted in a set of detectors, from which the model with the best A' was selected. A' is the probability that if the detector is comparing two students, with one student drawn from each of the two groups being classified, it will correctly identify which student is from which group (by determining which student has a higher predicted gaming percentage). A' is computed by considering all possible thresholds between two categories of students and looking at the proportion of true and false positives and negatives at each threshold. A' is equivalent to both the area under the ROC curve in signal detection theory, and to W, the Wilcoxon statistic (Hanley and McNeil 1982). A model with an A' of 0.5 performs at chance, and a model with an A' of 1.0 performs perfectly—a model with a reasonably good A' of 0.75 at distinguishing GAMED-HURT students from non-gaming students would, when given a GAMED-HURT student and a non-gaming student, be able to correctly select the

[2] In the implementation of Fast-Correlation Based Filtering used within the research presented here, linear correlation is used as the goodness-of-fit measure rather than entropy, as the overall model architecture is based on linear correlation.

GAMED-HURT student 75% of the time. A' was averaged across the model's ability to distinguish GAMED-HURT students from non-gaming students, and the model's ability to distinguish GAMED-HURT students from GAMED-NOT-HURT students.

Using linear correlation during the process of finding candidate models, and then using A' to select between candidate models, made it possible to find a model that was excellent on both metrics without needing to repeatedly calculate A' (at considerable time-cost) during the process of finding candidate models.

5 Validation

In this section of the paper, we will investigate how accurate the gaming detectors are at identifying those students who game the system in a way associated with poorer learning. Accuracy, in this case, is assessed in terms of generalizability. Two forms of generalizability are of interest: whether a detector trained on a population of students within a specific lesson effectively detects harmful gaming in new students using that lesson, and whether a detector trained within a specific lesson (or set of lessons) effectively detects harmful gaming in new lessons. Within this section, both types of generalizability are assessed.

5.1 Generalization to new students

We assessed the detector's ability to generalize to new students by conducting a Leave-One-Out-Cross-Validation (LOOCV) using data from the lesson that we had the most data for, the Scatterplot lesson. We took a gaming detector that had been fit using data from all 237 students who used the Scatterplot lesson (2003–2005), and re-fit the parameter values for each set of 236 students. In each case, we then used the re-fit parameter values to calculate the gaming frequency of the left-out 237th student. This gave us a set of 237 cross-validated predictions, one per student.

We will first consider the detector's performance without cross-validation, i.e. testing on the training set. We will assess the detector's performance in terms of its A' values. The difference between two A' values, or the difference between an A' value and chance, can be computed using the standard formula for the Z statistical test in combination with Hanley and McNeil's (1982) technique for estimating the standard error of an A' value. The Z distribution is very similar to the distribution used in the t statistical test for sample sizes over 30, and like the t value given by the t-test, a Z value can be converted to a two-tailed p-value; a Z value of 1.96 corresponds to a two-tailed p-value of 0.05.

When tested on the training set (all 237 students who used the Scatterplot lesson), the gaming detector achieved an A' value of 0.80 at distinguishing students who gamed the system and had poorer learning, from non-gaming students. This result was statistically significantly better than chance, $Z = 5.00$, two-tailed $p < 0.001$. The gaming detector also achieved an A' value of 0.71 at distinguishing between the two types of gaming behavior. This result was also significantly better than chance, $Z = 3.03$, two-tailed $p < 0.01$. It is worth noting, incidentally, that the gaming detector is not simply catching students who learn poorly, because it only achieves an

A' of 0.53 at distinguishing students who have poorer learning but who do not game the system, a result which is not statistically significantly different than chance, $Z = 0.57$, two-tailed $p = 0.57$.

When the gaming detector's generalizability was tested using Leave-One-Out-Cross-Validation, the detector achieved an A' value of 0.73 at distinguishing students who gamed the system and had poorer learning, from non-gaming students. This result was significantly better than chance, $Z = 3.39$, $p < 0.001$. The drop in performance from the training set to cross-validation, 0.80 to 0.73, was not statistically significant, $Z = 0.85$, two-tailed $p = 0.39$. Under LOOCV, the gaming detector also achieved an A' value of 0.68 at distinguishing between the two types of gaming behavior. This result was also significantly better than chance, $Z = 2.43$, two-tailed $p = 0.02$. The drop in performance from the training set to cross-validation, 0.71 to 0.68, was again not statistically significant, $Z = 0.37$, two-tailed $p = 0.71$.

Hence, the gaming detector is significantly better than chance, both when tested on the original training set, and when Leave-One-Out-Cross-Validation is used to test the detector's ability to transfer to new students. In addition, though there is some appearance of a degradation in performance when the detector is transferred to new students, that degradation is not statistically significant.

5.2 Generalization across tutor lessons

A detector that transfers to new populations of students, within a single tutor lesson, can be used as the basis for a system that responds to gaming the system, within that lesson. (cf. Baker et al. 2006; Walonoski and Heffernan 2006b). However, in order to be useful across large-scale, year-long tutor curricula (cf. Corbett et al. 2001; cf. van Lehn et al. 2005), a detector must be able to generalize across tutor lessons.

To determine whether our approach to gaming detection can generalize across multiple tutor lessons, we train the detector on three tutor lessons and then evaluate its performance when transferred to a fourth, left-out tutor lesson—doing so across each potential split of three training lessons and one test lesson. We will compare the accuracy of the detectors, when transferred to a new lesson, to the detectors' accuracy within the three lessons each detector was trained on.

In doing this, we will need to rely upon more complicated statistics than in the previous section, because the data from the different tutor lessons is partially, but not fully, independent (because in many cases, some but not all of the students used two of the tutor lessons studied). We use a meta-analytical technique, Strube's Adjusted Z (1985), in order to avoid overemphasizing the information from the students who used multiple tutor lessons. Strube's Adjusted Z explicitly incorporates inter-correlation between dependent measures (in this case, each student's observed gaming frequency in each lesson) into calculations of statistical significance, in order to avoid either making an overly-conservative estimate of statistical significance (such as in the mean Z technique, which assumes a correlation of 1 between dependent measures), or making an under-conservative estimate (such as in Stouffer's Z—Rosenthal and Rosnow, 1991—which assumes a correlation of 0 between dependent measures). Z-scores calculated using Strube's Adjusted Z are denoted $Z_{a,}$, and can be treated statistically in the same fashion as any Z-score (i.e. values of Z greater

Table 2 The performance of detectors trained on three of the four lessons, on training and test lessons

Lessons detector trained on	A' (GAMED-HURT vs NON-GAMING) when detector tested on lesson			
	Scatterplot	Percents	Geometry	Probability
Percents, Geometry, Probability	**0.67**	0.91	0.77	0.96
Scatterplot, Geometry, Probability	0.75	**0.86**	0.76	0.99
Scatterplot, Percents, Probability	0.81	0.93	**0.69**	0.92
Scatterplot, Percents, Geometry	0.75	0.92	0.77	**0.99**

All values in this table are statistically significantly higher than chance. Values applying to a detector's performance in a lesson it was not trained on are in bold

Table 3 The performance of detectors trained on three of the four lessons, on training and test lessons

Lessons detector trained on	A' (GAMED-HURT vs. GAMED-NOT-HURT) when detector tested on lesson			
	Scatterplot	Percents	Geometry	Probability
Percents, Geometry, Probability	**0.6**	0.8	0.92	0.99
Scatterplot, Geometry, Probability	0.69	**0.75**	0.94	0.99
Scatterplot, Percents, Probability	0.74	0.9	**0.84**	0.99
Scatterplot, Percents, Geometry	0.68	0.8	0.89	**0.99**

All values in this table are statistically significantly higher than chance. Values applying to a detector's performance in a lesson it was not trained on are in bold

than 1.96 correspond to $p < 0.05$). Full detail on Strube's adjusted Z is given in Appendix II.

The gaming detectors trained on three lessons achieve an average A' of 0.85 at distinguishing GAMED-HURT students from non-gamers, in the training lessons, and an average A' of 0.80 at making the same distinction in the test lessons, as shown in Table 2. Hence, the detectors seem to do a little better in the training lessons than in the test lessons, but the difference between the performance of these detectors, from training lessons to test lessons, was not statistically significant, $Z_a = 1.17$, $p = 0.24$.

These detectors achieve an average A' of 0.86 at distinguishing GAMED-HURT students from GAMED-NOT-HURT students, in the training lessons, and an average A' of 0.80 at making the same distinction in the test lessons, as shown in Table 3. Again, the detectors seem to do a little better in the training lessons than in the test lessons, but the difference between the performance of these detectors, from training lessons to test lessons, was not significant, $Z_a = 1.37$, $p = 0.17$.

It is worth noting, however, that the effects are somewhat unstable across different combinations of lessons, as shown in Tables 2 and 3. In some cases, transfer is almost

perfect—but in other cases, it is poorer. On the whole, transfer is successful inasmuch as there is not statistically significant degradation, but transfer does not appear completely even. It is likely that some characteristics which lessons may share are more likely to promote successful transfer, and that if lessons do not share these characteristics, transfer will be less successful. Determining which characteristics of a lesson are particularly important for transfer of behavior detectors, such as the gaming detector, is an important area for future research.

Overall, then, training on three lessons appears to result in a detector which transfers effectively to a new tutor lesson with at most a mild degradation in performance—suggesting that this detector is not capturing properties specific to gaming in individual lessons, but properties which are general to the overall Cognitive Tutor that these lessons were drawn from. The overall pattern of results is shown in Tables 2 and 3.

5.3 Is the detector accurate enough to use to drive interventions?

In the preceding sections, we have demonstrated that the detector can accurately predict the frequency of gaming for students that it was not trained on, and can even accurately predict performance within new lessons when trained on three lessons.

However, the fact that the detector is statistically significantly more accurate than chance does not immediately indicate that it will be sufficiently accurate to drive interventions within a learning environment.

It is open to debate exactly how effective a detector should be to drive appropriate interactions within an interactive system; in particular, less reliable detection can be managed by an interactive system through using "fail-soft" interventions (cf. Liu and Singh 2002) which do not lead to highly negative consequences if the detector is inaccurate.

Thus, the question of how effective a detector must be depends in part on the intervention it is used with; and therefore the best evidence that a detector is sufficiently accurate to drive interventions is the existence of a learning system which uses that detector to drive interventions, and where the interventions have positive consequences.

Such evidence exists for the gaming detector reported here in the form of Scooter the Tutor (Baker et al. 2006), an interactive agent incorporated into the Cognitive Tutor. Scooter responds to gaming in two fashions: through expressions of negative emotion when students game the system, and through offering supplementary exercises on steps which a student has gamed several times. Scooter significantly reduced the frequency of gaming, and significantly improved gaming students' learning. The success of this system suggests that the gaming detector, when used with an appropriate intervention (both of these interventions were relatively fail-soft, having relatively minimal consequences if the detector was inaccurate), is—if not perfect—sufficiently accurate to drive interventions.

While a full discussion of Scooter and the study evaluating its effectiveness are outside of the scope of this paper, interested readers are referred to (Baker et al. 2006).

6 Behavioral analysis

In this section, we will discuss what behaviors are captured by a detector of harmful gaming. Specifically, we will discuss the parameters that make up a detector of harmful gaming trained on all four tutor lessons. This detector's parameters are shown in Table 4. Within the detector, multiple parameters rely upon the same behavioral features, in different combinations, and thus some of the relationships between individual features and gaming are fairly complex. Hence, rather than discussing the detector's parameters one-by-one, we will discuss the relationships between student behavior and gaming that emerge from the detectors' parameters.

In the detector's characterization of harmful gaming, harmful gaming is associated with consistently making many errors on a specific problem step, across problems (Feature GH1 in Table 4). The evidence for harmful gaming is even stronger if the student gets the step right on the first try in some problems and makes a large number of errors in other problems (Feature GH2). However, not all errors are evidence of harmful gaming—for instance, errors made while point-plotting (an activity where slips are common—i.e. making an error despite knowing the skill well) are not associated with harmful gaming (Feature GH3).

Specific types of help use are also associated with harmful gaming; specifically, requesting help on several steps in short succession is considered gaming, once the student has achieved a high probability of knowing at least some steps (i.e. once the student has completed at least a couple of problems) (Feature GH4).

The relationship between the time taken to perform an action and harmful gaming is somewhat complex. Quick actions are evidence of gaming, but only if the student has already made an error on the current step (Feature GH5). However, extremely quick errors or help requests (more than 2 SD faster than normal—in many cases taking place in less than a fifth of a second) are not seen as evidence of gaming;

Table 4 The detector of harmful gaming (GH)

	Param 1	Param 2	Value	Description
GH1	howmanywrong	wrongpct	0.08	GH: Many errors across problems
GH2	pknow-direct	wrongpct	1.25	GH: History of many errors and yet a high probability the student knows the skill (i.e. lots of errors on some problems, other times correct on the first try)
GH3	point	wrongpct	−2.22	Not GH: Lots of errors while plotting points
GH4	pknow	recent8help	0.66	GH: Asking for a lot of help, and then reaching a step which the system knows the student knows
GH5	punchange	timelast3SD	−0.72	GH: Very fast actions after making at least one error
GH6	timelast3SD	timelast3SD	−0.34	Not GH: Very fast answers or very slow answers
GH7	timelast3SD	wrongpct	0.37	Not GH: Very fast answers on steps with a high frequency of errors across problems

In all cases, param1 is multiplied by param2, and then multiplied by value

this is because in many cases, actions at this speed consist of identical rapid actions such as accidental double-clicks on help or hitting enter twice (Features GH6 and GH7).

Overall, these behaviors appear at face-value to be a reasonable match to the types of behaviors the human observers were looking for. However, a number of behaviors could plausibly have seemed to match to a common-sense definition of gaming. The specific set of behaviors within the detector have met a stronger standard—in combination, these behaviors are a good statistical match to concrete observations of which students game and fail to learn.

7 Comparing the gaming detector to related systems

In this section, we compare the Gaming Detector our group has developed to systems developed by other research groups which also detect gaming behavior. We will first advance a list of criteria for an ideal detector of gaming behavior, and then consider both our Gaming Detector and other systems according to these criteria.

7.1 Criteria for an ideal detector of gaming behavior

A detector of gaming behavior can be developed with many goals in mind. In this section, we consider a set of potential criteria for evaluating a detector of gaming behavior, or detectors of student usage behavior in general.

First, we propose that an ideal detector should accurately identify a category (or categories) of behavior which is known to be associated with a meaningful difference in student experience or outcomes. Detecting a behavior which is associated with negative outcomes in the student's experiences or learning may make it possible to develop systems which can respond to student behavior in a way that concretely improves students' learning and/or experiences. Since gaming the system is known to be associated with poorer learning (Baker et al. 2004), any system that accurately identifies gaming behavior will satisfy this goal. If a system can go on to distinguish gaming behaviors that have relatively greater or lesser impact on learning, the system will have succeeded strongly at this goal.

Second, an ideal detector can predict not only which students engage in the behavior, but *when* they do so. If a system can effectively predict when a student is engaging in a relevant behavior, the system can present interventions just-in-time, making more types of interventions (and potentially more effective interventions) possible. Hence, any system that can predict *when* students game will satisfy this goal. A system succeeds strongly at this goal if those predictions are validated, either directly through comparing the predictions to knowledge about exactly when a student was gaming, or by showing that a system that uses the predictions to respond to gaming behavior does in fact reduce gaming behavior.

Third, an ideal detector not only detects student behaviors but can help researchers understand those behaviors better. Despite an increasingly rich history of research into learner–computer interaction, our understanding of why students choose certain behaviors when using learning environments is still quite limited. A detector

which indicates exactly when students engage in a behavior can potentially help us understand why students engage in that behavior. Hence, a system which has been used to understand gaming behavior better will satisfy this goal.

Fourth, an ideal detector can generalize. While a detector developed within the context of a small system may provide insights into detector development and student behavior, it will have less impact than a detector which can be applied more broadly. Thus far, most detectors of student behaviors have been developed using data from fairly small-scale learning environments, or using individual lessons from a larger curriculum. However, interactive learning environments such as intelligent tutors are increasingly being used as major components in semester or year-long curricula (Corbett et al. 2001; van Lehn et al. 2005). Therefore, to be widely useful—and used—detectors of behaviors and motivation will need to be generalizable beyond a single tutor lesson or small-scale system. A system which can generalize to new tutor lessons, or better yet, entirely new tutors, will satisfy this goal.

7.2 Other detectors of gaming behavior

Beyond the Gaming Detector which was the principal subject of this paper, at least five other systems can be considered detectors of gaming behavior.

The first of these systems, Aleven et al.'s (2004) Help-Seeking Tutor Agent, was first presented at the same conference that the Gaming Detector was first presented at (Intelligent Tutoring Systems 2004, in Brazil). The Help-Seeking Tutor Agent models a set of behaviors related to student help-seeking within Cognitive Tutors. Two of those behaviors, try-step abuse and hint abuse, correspond to the broader category of gaming the system. The Help-Seeking Tutor Agent was developed using knowledge engineering to develop the functional form of a mathematical model, and then automated parameter-fitting to find values for the parameters of that model. The Help-Seeking Tutor Agent has been validated to transfer to at least one new tutor lesson, after re-fitting parameter values on new data (Roll et al. 2005), and has been used as the basis of a learning system which gave students feedback on their metacognitive errors (Roll et al. 2007). This system improved students' metacognitive behavior during their use of the system, but did not affect students' domain learning or their metacognitive behavior in other contexts.

The second system is Beck's (2005) Disengagement Tracing. Disengagement Tracing models whether a student is responding faster on quiz items than would be possible if the student was sincerely attempting to answer the item using their knowledge, within Project LISTEN, an intelligent tutor teaching reading skills. Disengagement Tracing was developed using knowledge engineering to develop an item-response theory model, and then by using automated parameter-fitting to find values for the parameters of that model. Disengagement Tracing was applied to an entire year of data, and was found to be accurate at predicting student knowledge on a post-test.

The third system is Johns and Woolf's (2006) system to detect student motivation and proficiency. This system detects whether students are abusing hints or guessing, the same behaviors identified by the Gaming Detector and the Help-Seeking Tutor Agent, but within a different math tutoring system, Wayang Outpost. Johns and Woolf's system was developed using knowledge engineering to develop the functional form

of a mathematical model, and then automated parameter-fitting to find values for the parameters of that model. Their system was used to decide whether to present interventions to gaming students, and the interventions were associated both with reduced gaming behavior and improved learning (Arroyo et al. 2007).

The fourth system is Walonoski and Heffernan's (2006a) Off-Task Gaming Behavior Detector. This system detects whether a student is gaming the system while using ASSISTments, an intelligent tutor which both tutors students on mathematics topics and informs their teacher as to which parts of the state mathematics exams may be giving the student particular difficulty. Walonoski and Heffernan's system, like the Gaming Detector, was developed using machine learning on human observations of the frequency of gaming behavior, and accurately correlated to the observations. The Off-Task Gaming Behavior Detector was used as the basis of visualization-based interventions which significantly reduced the frequency of gaming behavior as well as reducing students' gaming in future tutor lessons (Walonoski and Heffernan 2006b).

Finally, the fifth system is Beal et al.'s (2006) model of student strategies. This system can predict whether students are engaging in either help abuse or guessing, while using Wayang Outpost, the same system studied by Johns and Woolf. Beal et al.'s model was developed by knowledge engineering, with parameters set in advance rather than through parameter-fitting. Beal and her colleagues clustered students by their adoption of a variety of strategies, including the gaming strategies, and found that including estimates of gaming frequency improved the accuracy of learner modeling (Beal et al. 2007).

7.3 Comparing detectors of gaming behavior

7.3.1 Accurately identifying which students game

Each of the six detectors of gaming behavior appear to satisfy the first criterion, accurately identifying gaming the system, a category of behavior known to be associated with significantly poorer learning—though one system is only indirectly validated. Interestingly, the six detectors are validated in fairly different ways. Two systems, the Gaming Detector and the Off-Task Gaming Behavior Detector, validate by comparing detector predictions to labels created by researchers. Three systems, Disengagement Tracing, Johns and Woolf's system, and Beal et al.'s system, attempt to validate by showing that incorporating their detectors improves prediction of student problem-solving performance within their system. In Disengagement Tracing and Beal et al.'s system, the validation is successful—in the case of Johns and Woolf's system, the validation is not a success; their system does not significantly improve prediction of student correctness. Four systems, the Gaming Detector, Disengagement Tracing, the Help-Seeking Tutor Agent, and the Off-Task Gaming Behavior Detector, also validate through correlating the predictions of gaming behavior to the student's performance on knowledge measures external to the learning system (either pre-tests or post-tests).

Five of six systems have thus been successfully validated in some fashion—the sixth, Johns and Woolf's system—has not yet been directly validated. However, its

predictions about gaming behavior have been used to drive interventions that appear both to reduce gaming behavior (as measured by the system) and improve learning. Therefore, we can consider Johns and Woolf's system indirectly validated.

Each of these systems detects gaming behavior. The Gaming Detector, alone among the six systems, goes a step further and distinguishes between types of gaming behavior associated with different learning outcomes. It is not immediately clear why only the Gaming Detector makes this division. Since the Gaming Detector's division between harmful and non-harmful gaming was made due to serendipity in early stages of machine learning, one possibility is that the four efforts that used knowledge engineering instead of machine learning did not find a separation in gaming behavior because they never looked for it, nor created the possibility that it could be found through serendipity. This does not explain, however, why Walonoski and Heffernan, who also used machine learning in developing the Off-Task Gaming Behavior Detector, did not also serendipitously discover the split in gaming behaviors as was found in the development of the Gaming Detector. Since none of the other five research groups explicitly looked for a split in gaming behavior (or at least did not report doing so), it remains an open question whether this split, replicated across lessons for the Gaming Detector, is unique to Cognitive Tutors or is a more widely generalizable finding. Nonetheless, at least at this point, the Gaming Detector may satisfy this first criterion more strongly than the other systems, by not just detecting gaming behavior but by distinguishing between types of gaming behavior.

7.3.2 Accurately identifying when students game

All six detectors of gaming behavior also satisfy the second criterion, predicting not only which students engage in gaming but when they do so. Four of the six systems—the Gaming Detector, the Help-Seeking Tutor Agent, Johns and Woolf's system, and the Off-Task Gaming Behavior Detector—go a step further and use these predictions to drive interventions that respond to gaming behavior. In all four cases, the interventions change the frequency of gaming behavior within the system. Varying effects are seen on student learning, but this is quite likely due to the differences between the interventions rather than the detection systems. In the specific case of the Gaming Detector, interventions driven by the Gaming Detector's predictions both significantly reduced the degree of gaming behavior (measured by live observations) and improved gaming students' learning (Baker et al. 2006). Walonoski and Heffernan's (2006a) Off-Task Gaming Behavior Detector goes a step further still—they probabilistically match the times of their observations to the actions in their tutor log files, and find that their detector's predictions match (in time) to the observations. As not all of the observations used to train the Gaming Detector in various contexts are synchronized with the tutor logs as well as Walonoksi and Heffernan's observations were, this direct validation is not currently possible for the Gaming Detector. Hence, three systems have indirect validation of this criterion; Walonoski and Heffernan's system is the only one to directly validate it.

7.3.3 Increasing our knowledge about gaming

Four of the six detectors of gaming behavior satisfy the third criterion, having been used to increase our knowledge of the construct of gaming. The Gaming Detector suggests that gaming may have different effects on learning depending on when students engage in gaming. In addition, the Gaming Detector tells us that gaming occurs in clusters, but that those clusters are not solely temporal; instead, those clusters take place both in specific pockets of time and also across time, linked to a specific problem step. This suggests that gaming may be a specific response to specific material rather than an overall response to the interactive learning environment. Thirdly, the Gaming Detector has been used to identify which student attitudes and motivations are associated with gaming, by comparing its predictions to student questionnaire responses (Baker et al. 2005, in press).

The Off-Task Gaming Behavior Detector and Beal et al.'s model have also been used in this fashion, to identify which student attitudes and motivations are associated with gaming, by comparing its predictions to student questionnaire responses (Baker et al. in press; Beal et al. 2006). Beck's Disengagement Tracing has been applied to an entire year of data, and shows that gaming behavior increases during the course of the year, and that gaming occurs in temporal clusters, confirming the Gaming Detector's finding of temporal clusters. Hence, four of the six systems have been used to increase our knowledge of the construct of gaming. It is likely that the other two systems ***could*** be used to understand more about gaming—however, the fact that they have not yet been used in this fashion (to the best of our knowledge), restricts them for now from having fulfilled this criterion.

7.3.4 Generalizing across students and contexts

Finally, all of the detectors of gaming behavior satisfy the fourth criterion, generalizability, but to different degrees. All six systems work when transferred to new students, using cross-validation or a new test set.

Beyond this, Disengagement Tracing, the Off-Task Gaming Behavior Detector, Johns and Woolf's system, and Beal et al.'s system, all successfully work within fairly large-scale learning environments. However, none of these four systems have been validated to transfer to new tutor contexts (i.e. new units/lessons with even moderately different interfaces or pedagogy). The Help-Seeking Tutor Agent has been shown to effectively transfer to a new tutor lesson without changing the functional form, but with re-fitting of parameter values (Roll et al. 2005). Only the Gaming Detector has been validated to successfully generalize to new lessons within the tutor curriculum, with no re-training.

7.3.5 Summary

In the previous four sections, we have discussed six systems that attempt to detect gaming behavior in terms of four criteria, accurately identifying which students game, accurately identifying when students game, increasing our knowledge about gaming, and generalizing beyond the original training context.

All six systems are at least acceptable on at least three of the four criteria. The Gaming Detector achieves the highest mark (alone, or in a tie) on three of the four criteria. Only in one case—the second criterion—does another system achieve the highest mark, and in that case the Gaming Detector is tied for second place. Hence, the approach used to develop the Gaming Detector appears to be capable of producing behavior detection systems which provide excellent performance on each of the four criteria of interest. A summary of the systems' performance is given in Table 5.

8 Conclusions

In this paper, we have presented a system that can detect when students "game the system", in a fashion associated with poor learning. We have shown that this detector

- accurately detects which students game the system
- makes predictions about *when* students game the system, which are difficult to directly validate, but which have been used to drive learning interventions which improve student learning
- expands our knowledge about the behavioral construct of "gaming the system"
- can generalize between contexts

Since the advent of this detector, multiple systems have been produced which also attempt to detect gaming behavior. These systems use a variety of techniques, and have largely been successful at achieving many of the same goals as our system. The Help-Seeking Tutor Agent and Gaming Detector have even been applied to the same data set, and the two detectors have been found to correlate well to one another (Roll et al. 2005).

On the whole, the existence of several different systems which can detect gaming behavior indicates that gaming is a fairly robust construct, present across many intelligent tutoring systems. However, it is not yet clear to what degree some features of gaming behavior which we have discovered—such as the split between harmful and non-harmful gaming—are general across intelligent tutoring systems.

Though gaming has been documented in many interactive learning contexts beyond intelligent tutors (cf. Magnussen and Misfeldt 2004; Cheng and Vassileva 2005), thus far all gaming detectors have been developed within the context of intelligent tutoring systems. An interesting and valuable area of future work will be to study the development of gaming detectors for these contexts, and how lessons learned in the development of gaming detectors within intelligent tutoring systems can transfer to these domains.

Beyond the future development of gaming detectors in other domains, the work to develop detectors of gaming behavior may facilitate the development of detectors of other types of behavior. Recently, the first author of this paper developed a system which can automatically detect whether a student who is idle when using an intelligent tutoring system is off-task or asking the teacher or another student for help (Baker 2007). This detector was developed using the exact same model framework as the system presented in this paper, and uses a model selection algorithm very similar to the one used here. Hence, the tools used within this paper—Latent Response Models

Table 5 Assessing all six systems which detect gaming behavior, across four criteria

System (first report of system)	Identifies which students game	Predicts when students game	Has increased knowledge about gaming construct	Has been shown to generalize
Gaming Detector (2004)	**Yes—correlates to both observations and external knowledge test. Plus, identified sub-categories within overall construct**	Yes, and those predictions have been used to drive interventions	**Yes**	**Yes, successfully transfers to new contexts with no re-training**
Help-Seeking Tutor Agent (2004)	Yes, correlates to external test of knowledge	Yes, and those predictions have been used to drive interventions	Not yet	Yes, successfully transfers to new contexts with only parameter re-fitting
Disengagement Tracing (2005)	Yes, predicts behavior within system and correlates to external test of knowledge	Yes	**Yes**	Perhaps—works within large-scale system
Johns and Woolf (2006)	Failed to improve predict of student behavior within system; however, used in effective intervention	Yes, and those predictions have been used to drive interventions	Not yet	Perhaps—works within large-scale system
Off-Task Gaming Behavior Detector (2006)	Yes, correlates to both observations and external knowledge test	**Yes, and those predictions have been validated and used to drive interventions**	**Yes**	Perhaps—works within large-scale system
Beal et al. (2006)	Yes, accurately predicts student behavior within system	Yes	**Yes**	Perhaps—works within large-scale system

Boldface indicates that a given system is best (or tied for best) on a given criterion

and Fast Correlation-Based Filtering—seem to be tools which can be applied to modeling a variety of user behaviors in interactive learning environments. Meta-analysis, though not yet widely used for analyzing user models, is another type of technique with considerable potential for our community. Meta-analysis provides a relatively easy-to-use pool of techniques for comparing models across different contexts, or aggregating measures of model accuracy across contexts.

As detectors of behavior categories such as gaming the system are developed for more systems, they may also provide leverage for developing detectors of other constructs, such as affect (cf. D'Mello et al. in press). Recent research has suggested that gaming the system co-occurs with some affective states (frustration, boredom, confusion) and that students who game the system are more likely to be bored in the future (Rodrigo et al. 2007). Hence, accurate gaming detection may support more accurate detection of student affect; correspondingly, the development of accurate detectors of student affect may increase the accuracy of future systems which detect gaming the system.

We close with a final thought on the broader generalizability not of this detector, or even the methods used to develop or evaluate it, but of the ideas it represents. Though the domain of this paper has fallen within the area of educational interactions, it is worth noting that many of the same issues apply in general to the problems of modeling user behavior and strategies. Detectors of user behavior and strategies should focus on behaviors which are associated with differences in user experience and outcomes. A detector of a category of user behavior should not just identify that a behavior has occurred, but when it occurs—as in the example presented within this paper, this task may be facilitated by using hierarchical modeling frameworks that make predictions at multiple grain-sizes. Detectors of user behavior and strategies will be more useful if they can help identify why users engage in the studied behaviors and strategies. Finally, detectors of user behaviors and strategies will be more useful if they can effectively generalize to different sub-domains within an overall system; and as in our example, training on broadly sampled data may result in a detector which can be applied more widely. Ultimately, a detector of behavior will be useful if it captures important behaviors, identifies when they occur, promotes understanding of the behaviors, and is general—regardless of what domain the behavior occurs within.

Appendix I

Gaming Detector Training Algorithm

Goal: Find model with good correlation to observed data, and good A'

Preset values:

π-The percentage of the best path's goodness-of-fit that is acceptable as an alternate path during fast correlation-based filtering (value used = 60%)

μ-The maximum acceptable correlation between a potential path's most recently added parameter and any alternate parameter with a better goodness-of-fit. (value used = 0.7)

ζ-The maximum size for a potential model (-1 if LOOCV is used to set model size). (value used = 7)

Data format:
A candidate model is expressed as two arrays: one giving the list of parameters used, and the second giving each parameter's coefficient.
Prior Calculation Task: Find correlations between different parameters
For each pair of parameters,
Compute linear correlation between the pair of parameters, across all actions, and store in an array
Main Training Algorithm:
Set the number of parameters currently in model to 0
Set the list of candidate models to empty
MODEL-STEP (empty model)
For each candidate model (list populated by MODEL-STEP)
Calculate that model's A' value (for both GAMED-HURT versus NON-GAMING, and GAMED-HURT versus GAMED-NOT-HURT)
Average the two A' values together
Output the candidate model with the best average A'.
Recursive Routine MODEL-STEP: Conduct a step of model search
Input: current model
If there is at least one parameter already in the model,
Subgoal: Complete exploration down the current path
Create variable PREV-GOODNESS; initialize to -1.
Create variable CURRENT-GOODNESS; initialize to -1
Create array BEST-RECENT-MODEL
Repeat
For each parameter not already in the model
Use iterative gradient descent to find best model that includes both the current model and the potential parameter (using linear correlation to the observed data as the goodness of fit metric).
Store the correlation between that model and the data
Add the potential parameter with the best correlation to the model
If $\zeta = -1$ (i.e. we should use cross-validation to determine model size)
Create an blank array A of predictions (of each student's game freq)
For each student S in the data set
Use iterative gradient descent to find best parameter values for the current model, without student S
Put prediction for student S, using new parameter values, into array A
Put the linear correlation between array A and the observed data into variable CURRENT-GOODNESS
If CURRENT-GOODNESS > PREV_GOODNESS
PREV_GOODNESS = CURRENT-GOODNESS
Put the current model into BEST-RECENT-MODEL
Else
Put the current model into BEST-RECENT-MODEL
Until (the model size = ζ OR PREV_GOODNESS > CURRENT- GOODNESS)
Add BEST-RECENT-MODEL to the list of candidate models

Else
Subgoal: Select a set of paths
Mark each parameter as POTENTIAL
For each model parameter
Use iterative gradient descent to find best model consisting only of this potential parameter (using linear correlation to the observed data as the goodness of fit metric).
Store the correlation between that model and the data
Repeat
Find the parameter P whose associated candidate model has the highest linear correlation to the observed data (among the set POTENTIAL)
Mark parameter P as SEARCH-FURTHER
For all potential parameters Q marked POTENTIAL
If the linear correlation between parameter Q and parameter P is greater than μ, mark parameter Q as NO-SEARCH
If the linear correlation between the model with parameter Q and the observed data, divided by the linear correlation between the model with parameter P and the observed data, is less than π, mark parameter Q as NO-SEARCH
Until no more parameters are marked POTENTIAL
For each parameter R marked as SEARCH-FURTHER
Set N to the best model that has only parameter R (this has already been computed)
Recurse MODEL-STEP (N)

Appendix II

Strube's (1985) Adjusted Z

Strube's adjusted Z (1985) is a technique used when data in different data sets is partially, but not fully, independent. In this paper, this condition is true because some but not all of the students used two of the tutor lessons studied (scatterplot and geometry).

Strube's Adjusted Z explicitly accounts for the intercorrelation between data sets with overlapping participants. Strube's Adjusted Z is recommended in Rosenthal and Rosnow (1991) for precisely this statistical situation, and has been used in at least 20 papers in published scientific literature (Google Scholar, visited September 3, 2007). In this section, we provide a brief discussion of Strube's formula, adapted from Strube's text.

Strube's Adjusted Z generalizes Stouffer's formula for computing Z scores (cf. Rosenthal and Rosnow 1991) for the case where there is non-independence between samples. As Strube (1985) discusses, Stouffer's formula for Z corresponds to a linear combination of Z values, divided by the linear combination's standard deviation. The variance of a linear combination (the standard deviation squared) can be written as the sum of the variances of the individual terms plus their covariances. Stouffer's formula assumes independence and thus eliminates the covariance terms;

Strube's formula explicitly includes the covariance terms—the covariances between the measures which we have Z values for. In addition, where there is only partial overlap in population, the covariance terms can be treated as a combination of the inter-dataset correlation where individual variable values are taken from the overlapping students, but the total number of students in both data sets is treated as the N (this is not strictly Strube's formula, but is a clear extension of the formula, as non-overlapping students are independent of one another).

To give a brief example (using simplified numbers rather than the exact numbers from the data in this paper):

Let us say that we are combining two tests of whether a detector is statistically significantly better than chance, involving two overlapping data sets (which we will call the Lesson-X and Lesson-Y 2004 data sets). The significance of the detector's effectiveness in Lesson-X is $A' = 0.80$, $Z = 2.00$, and the significance of the detector's effectiveness in Lesson-Y is $A' = 0.85$, $Z = 2.50$. 75% of the students in the two lessons are represented in both lessons, and the other 25% only contributed data to one of the two lessons. Within the 75% of students represented in both lessons, the correlation between the observed gaming frequency in the two lessons is 0.25.

Stouffer's Z would calculate the combined Z as (2.00+2.50)/sqrt(num-tests) = 3.18 (there are two statistical tests here, hence num-tests = 2). However, this assumes full independence, which is in this case an overly liberal estimation (i.e. there is an overly high chance of Type II error).

Assuming full non-independence (the average Z method) would calculate the combined Z as (2.00 + 2.50)/(num-tests) = 2.25. However, since there is only partial non-independence, this estimate is overly conservative (i.e. there is an overly high chance of Type I error).

Using Strube's method, and treating non-overlapping students as independent from one another, we calculate Z as 2.00+2.50/sqrt(num-tests+0.25*0.75+0.25*0.75) = 2.92. This gives us a value between these two estimates, which is not biased towards either Type I or Type II error.

Acknowledgements This research was supported by an NDSEG (National Defense Science and Engineering Graduate) Fellowship, a fellowship from the Learning Sciences Research Institute at the University of Nottingham, and by IERI grant number REC-043779 to "Learning-Oriented Dialogue in Cognitive Tutors: Towards a Scalable Solution to Performance Orientation". We would like to thank Angela Wagner, Jay Raspat, Meghan Naim, Katy Getman, Pat Battaglia, Dina Crimone, Russ Hall, and Sue Cameron for assisting in the collection of the data discussed here. We would also like to thank Darren Gergle, Vincent Aleven, Dave Andre, Joseph Beck, and the anonymous reviewers for helpful discussions and suggestions.

References

Aleven, V.: Helping students to become better help seekers: towards supporting metacognition in a cognitive tutor. Paper presented at German-USA Early Career Research Exchange Program: Research on Learning Technologies and Technology-Supported Education. Tubingen, Germany (2001)

Aleven, V., McLaren, B.M., Roll, I., Koedinger, K.R.: Toward tutoring help seeking: applying cognitive modeling to meta-cognitive skills. In: Proceedings of the 7th International Conference on Intelligent Tutoring Systems (ITS 2004), pp. 227–239. Maceió, Brazil (2004)

Anderson, J.R., Corbett, A.T., Koedinger, K.R., Pelletier, R.: Cognitive tutors: lessons learned. J. Learn. Sci. **4**(2), 167–207 (1995)

Arroyo, I., Woolf, B.: Inferring learning and attitudes from a Bayesian Network of log file data. In: Proceedings of the 12th International Conference on Artificial Intelligence in Education, pp. 33–40. Amsterdam (2005)

Arroyo, I., Ferguson, K., Johns, J., Dragon, T., Meheranian, H., Fisher, D., Barto, A., Mahadevan, S., Woolf. B.P.: Repairing disengagement with non-invasive interventions. In: Proceedings of the 13th International Conference on Artificial Intelligence in Education, pp. 195–202. Marina del Rey, CA (2007)

Baker, R.S., Corbett, A.T., Koedinger, K.R.: Detecting student misuse of intelligent tutoring systems. In: Proceedings of the 7th International Conference on Intelligent Tutoring Systems, pp. 531–540. Maceió, Brazil (2004)

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". In: Proceedings of ACM CHI 2004: Computer-Human Interaction, pp. 383–390. Vienna, Austria (2004)

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.: Adapting to when students game an intelligent tutoring system. In: Proceedings of the 8th International Conference on Intelligent Tutoring Systems, pp. 392–401. Jhongli, Taiwan (2006)

Baker, R.S.J.d., Walonoski, J.A., Heffernan, N.T., Roll, I., Corbett, A.T., Koedinger, K.R.: Why students engage in "Gaming the System" behavior in interactive learning environments. To appear in J. Interact. Learn. Res. (in press)

Beal, C.R., Mitra, S., Cohen, P.R.: Modeling learning patterns of students with a tutoring system using Hidden Markov Models. In: Proceedings of the 13th International Conference on Artificial Intelligence in Education, pp. 238–245. Marina del Rey, CA (2007)

Beal, C.R., Qu, L., Lee, H.: Classifying learner engagement through integration of multiple data sources. In: Proceedings of the 21st National Conference on Artificial Intelligence, pp. 2–8. Boston (2006)

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

Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge, UK (2004)

Cheng, R., Vassileva, J.: Adaptive reward mechanism for sustainable online learning community. In: Proceedings of the 12th International Conference on Artificial Intelligence in Education, pp. 152–159. Amsterdam (2005)

Conati, C., McLaren, H.: Data-driven refinement of a probabilistic model of user affect. In: Proceedings of the Tenth International Conference on User Modeling (UM2005), pp. 40–49. Edinburgh, Scotland (2005)

Corbett, A.T., Anderson, J.R.: Knowledge tracing: modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction **4**, 253–278 (1995)

Corbett, A.T., Koedinger, K.R., Hadley, W.H.: Cognitive tutors: from the research classroom to all classrooms. In: Goodman, P.S. (ed.) Technology Enhanced Learning: Opportunities for Change, pp. 235–263. Lawrence Erlbaum Associates, Mahwah, NJ (2001)

D'Mello, S.K., Craig, S.D., Witherspoon, A.W., McDaniel, B.T., and Graesser, A.C.: Automatic detection of learner's affect from conversational cues. To appear in User Modeling and User-Adapted Interaction. (2008) DOI 10.1007/s11257-007-9037-6

de Vicente, A. and Pain, H.: Informing the detection of the students' motivational state: an empirical study. In: Proceedings of the Sixth International Conference on Intelligent Tutoring Systems, pp. 933–943. Biarritz, France (2002)

Hanley, J.A., McNeil, B.J.: The meaning and use of the area under a Receiver Operating Characteristic (ROC) curve. Radiology **143**, 29–36 (1982)

Johns, J., Woolf, B.: A dynamic mixture model to detect student motivation and proficiency. In: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06), pp. 163–168. Boston (2006)

Koedinger, K.R.: Toward evidence for instructional design principles: Examples from Cognitive Tutor Math 6. In: Proceedings of PME-NA XXXIII (the North American Chapter of the International Group for the Psychology of Mathematics Education), pp. 21–49. Athens, GA (2002)

Liu, H., Singh, P.L.: MAKEBELIEVE: using commonsense knowledge to generate stories. In: Proceedings of the 18th National Conference on Artificial Intelligence, AAAI 2002, pp. 957–958. Edmonton, Canada (2002)

Magnussen, R., and Misfeldt, M.: Player transformation of educational multiplayer games. In: Proceedings of Other Players. Copenhagen, Denmark. Available at http://www.itu.dk/op/proceedings.htm (2004). Accessed 13 May 2007.

Maris, E.: Psychometric latent response models. Psychometrika **60**(4), 523–547 (1995)

Mostow, J., Aist, G., Beck, J., Chalasani, R., Cuneo, A., Jia, P., Kadaru, K.: A La Recherche du Temps Perdu, or As Time Goes By: where does the time go in a Reading Tutor that listens? In: Proceedings of the Sixth International Conference on Intelligent Tutoring Systems (ITS'2002), pp. 320–329. Biarritz, France (2002)

Murray, R.C., vanLehn, K.: Effects of dissuading unnecessary help requests while providing proactive help. In: Proceedings of the 12th International Conference on Artificial Intelligence in Education, pp. 887–889. Amsterdam (2005)

Ramsey, F.L., Schafer, D.W.: The Statistical Sleuth: A Course in Methods of Data Analysis. Duxbury Press, Belmont, CA (1997)

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. In: Proceedings of the 13th International Conference on Artificial Intelligence in Education, pp. 145–152. Marina del Rey, CA (2007)

Roll, I., Baker, R.S., Aleven, V., McLaren, B.M., Koedinger, K.R.: Modeling students' metacognitive errors in two intelligent tutoring systems. In: Proceedings of User Modeling 2005, pp. 379–388. Edinburgh, Scotland (2005)

Roll, I., Aleven, V., McLaren, B.M., and Koedinger, K.R.: Can help seeking be tutored? Searching for the secret sauce of metacognitive tutoring. In; Proceedings of the 13th International Conference on Artificial Intelligence in Education, pp. 203–210. Marina del Rey, CA (2007)

Rosenthal, R., Rosnow, R.: Essentials of Behavioral Research: Methods and Data Analysis. McGraw-Hill, Boston, MA (1991)

Schofield, J.W.: Computers and Classroom Culture. Cambridge University Press, Cambridge, UK (1995)

Strube, M.J.: Combining and comparing significance levels from nonindependent hypothesis tests. Psychol. Bull. **97**, 334–341 (1985)

Tait, K., Hartley, J., Anderson, R.C.: Feedback procedures in computer-assisted arithmetic instruction. Br. J. Educ. Psychol. **43**, 161–171 (1973)

van Lehn, K., Lynch, C., Shulze, K., Shapiro, J.A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., Wintersgill, M.: The Andes physics tutoring system: Five years of evaluations. In: Proceedings of the 12th International Conference on Artificial Intelligence in Education, pp. 678–685. Amsterdam (2005)

Walonoski, J.A., Heffernan, N.T.: Detection and analysis of off-task gaming behavior in intelligent tutoring systems. In: Proceedings of the 8th International Conference on Intelligent Tutoring Systems, pp. 382–391. Jhongli, Taiwan (2006a)

Walonoski, J.A., Heffernan, N.T.: Prevention of off-task gaming behavior in intelligent tutoring systems. In: Proceedings of the 8th International Conference on Intelligent Tutoring Systems, pp. 722–724. Jhongli, Taiwan (2006b)

Wood, H., Wood, D.: Help seeking, learning, and contingent tutoring. Comp. Educ. **33**, 153–169 (1999)

Yu, L., Liu, H.: Feature selection for high-dimensional data: a fast correlation-based filter solution. In: Proceedings of the International Conference on Machine Learning, pp. 856–863. Washington, DC (2003)

Authors' vitae

Dr. Ryan S. J. d. Baker is a Post-Doctoral Fellow in the Human-Computer Interaction Institute and Pittsburgh Science of Learning Center at Carnegie Mellon University. Dr. Baker received his Sc.B. in Computer Science from Brown University and Ph.D. in Human-Computer Interaction from Carnegie Mellon University in 2005. His research focuses on the development of interactive learning environments that can adapt effectively and sensitively to differences in students' choices, affect, and motivation. He uses methods from user modeling, educational data mining, machine learning, and quantitative observation.

Dr. Albert T. Corbett is Associate Research Professor in the School of Computer Science at Carnegie Mellon University. Dr. Corbett received his B.A. in Psychology from Brown University and Ph.D. in Psychology from the University of Oregon. His primary research interests are in human memory, comprehension and problem solving and he has brought these interests to the design and evaluation of intelligent computer

tutors. These tutors have proven to be rich environments for pursuing basic research in human cognition that has applied significance.

Ido Roll is a Ph.D. candidate in Human-Computer Interaction at Carnegie Mellon University, and a member of the Program in Interdisciplinary Education Research (PIER) and the Pittsburgh Science of Leaning Center (PSLC). He received his B.Sc. in Mathematics and Physics from the Hebrew University of Jerusalem in 1995. His research focuses on making students more eager, capable, independent, and innovative learners, using Intelligent Learning Environments. More specifically, he is interested in improving students' usage of available help facilities, and in combining the benefits of structured discovery tasks with explicit instruction.

Dr. Kenneth R. Koedinger is Professor of Human-Computer Interaction and Psychology at Carnegie Mellon University, and the CMU director of the Pittsburgh Science of Learning Center (PSLC). He received his B.S. in Mathematics and Computer Science from the University of Wisconsin, and received his Ph.D. in Cognitive Psychology from Carnegie Mellon University. His research goal is to create educational technologies that dramatically increase student achievement. He has developed Cognitive Tutors, rich problem solving environments which provide just-in-time learning assistance, for mathematics and science, and has tested them in the laboratory and the classroom.