Educational Software Features that Encourage and Discourage "Gaming the System"

Ryan S.J.d. BAKER[1], Adriana M.J.B. DE CARVALHO, Jay RASPAT, Vincent ALEVEN, Albert T. CORBETT, and Kenneth R. KOEDINGER
Human-Computer Interaction Institute, Carnegie Mellon University

Abstract. Gaming the system, attempting to succeed in an interactive learning environment by exploiting properties of the system rather than by learning the material (for example, by systematically guessing or abusing hints), is prevalent across many types of educational software. Past research on why students choose to game has focused on student individual differences. Many student individual differences, including attitudes towards mathematics, have been shown to be associated with gaming, but generally with low correlation. In this paper, we investigate how individual differences between learning environments can increase or decrease the probability of gaming. We enumerate ways intelligent tutor lessons vary from each other, and use data mining to discover hypotheses about how differences in software design and content influence the choice to game the system. We discover a set of tutor features that explain 56% of the variance in gaming, over five times the degree of variance explained in any prior study of student individual differences and gaming. These results provide an important step towards developing prescriptions for designing intelligent tutor software that students game significantly less.

Keywords. Gaming the System, Educational Data Mining, Intelligent Tutoring System, Software Features.

Introduction

In recent years, there has been considerable progress towards designing educational software that can respond in appropriate ways when students game the system [4], attempting to succeed in an interactive learning environment by exploiting properties of the system rather than by learning the material (by systematically guessing or abusing hints). These systems respond to gaming in a variety of ways, including the provision of supplementary exercises [3], meta-cognitive messages [1, 16], and visualizations of a student's degree of gaming behavior [17]. These strategies have achieved partial success at reducing gaming and mitigating its effects, albeit at the cost of making student-computer interaction considerably more complex. This approach for responding to gaming emerged from two factors – first, the failure of

[1] Corresponding Author. This work was funded by NSF grant REC-043779 to "IERI: Learning-Oriented Dialogs in Cognitive Tutors: Toward a Scalable Solution to Performance Orientation", and by the Pittsburgh Science of Learning Center, National Science Foundation award SBE-0354420.

approaches designed to eliminate gaming by making it harder to game (students found new ways to game) [e.g. 14], and second, the belief that individual differences between students, such as goal orientation, attitudes towards mathematics, or passive-aggressiveness, explained most of the variance in gaming behavior [cf. 2, 4, 7, 8].

However, thus far, the evidence seems weak for the belief that individual differences are the major factor explaining gaming, with several studies reporting statistically significant relationships, but accounting for fairly little of the variance in how much gaming occurred. Arroyo and Woolf [2] reported an analysis of associations between student attitudes and gaming, reporting associations explaining around 9% of the variance in gaming ($r^2 = 0.09$). A collaboration between researchers at WPI and CMU [7] reported three additional studies along these lines, testing 13 hypotheses; no individual difference accounted for more than 5% of the variance in how much a student gamed ($r^2 < 0.05$). Beal et al. [8] reported a study investigating 5 hypotheses, and found some student characteristics to be significantly associated with gaming; however, this paper did not report correlations between gaming and these characteristics, making it uncertain whether the associations were stronger than those reported in these earlier papers.

Baker [5] attempted to determine if this pattern of results arose from researchers investigating the wrong individual differences, or if another type of factor – aspects of the software itself – might explain more of the variance in gaming behavior. He compared student-level and software-level explanations as classes (i.e. statistically considering the overall category of explanation rather than individual explanations), and found that differences in tutor lessons, as classes, explained over three times as much of the variance in gaming as student individual differences did.

In this paper, we present a study which follows up on this finding, aiming to discover *which* aspects of educational software predict gaming. In this study, data was obtained from the PSLC DataShop [13] for an entire school year of the use of Cognitive Tutor Algebra. During the school year, students worked through a variety of lessons on different topics, with moderate variation in subject matter and considerable variation in design, making it possible to observe which differences in subject matter and/or design are associated with differences in how much gaming occurs. A taxonomy of the differences between tutor lessons was developed, and this taxonomy was used to determine which lesson features are most strongly associated with increased gaming.

1. Data

Data was obtained from the PSLC DataShop (dataset: Algebra I 2005-2006 Hampton Only) [13], for 58 students' use of Cognitive Tutor Algebra during an entire school year. The data set was composed of approximately 437,000 student transactions (entering an answer or requesting help) in the tutor software. All of the students were enrolled in algebra classes in one high school in the Pittsburgh suburbs which used Cognitive Tutors two days a week, as part of their regular mathematics curriculum. None of the classes were composed predominantly of gifted or special needs students. The students were in the 9^{th} and 10^{th} grades (approximately 14-16 years old).

The Cognitive Tutor Algebra curriculum involves 32 lessons, covering a complete selection of topics in algebra, including formulating expressions for word problems, equation solving, and algebraic function graphing. Data from 10 lessons was eliminated from consideration, due to having insufficient data to be able to conduct at least 498

observations, discussed below (500 was the planned cut-off, but a lesson with 498 observations was included). On average, each student completed 9.9 tutor lessons (among the set of 22 lessons considered), for a total of 577 student/lesson pairs.

To determine how often each student gamed the system, in each lesson, each student's actions were retrospectively labeled using "text replays" [6]. In text replays, a segment of student behavior from log files is shown in pretty-printed form, and the coder identifies the segment, in terms of whether it involves the behavior category of interest (in this case, gaming). Text replays are fast to conduct, and achieve acceptable inter-rater reliability – Cohen's κ=0.58 in one study involving labeling gaming the system [6]. One coder (the second author) made 18,737 text replay observations across the 22 units (labeling approximately a quarter of the transactions in the entire data set), in just over 200 hours, after multiple training sessions and checks of labeling agreement with the first author. The segments of the log files displayed were chosen by stratified sampling, across lessons and students, in order to achieve a comparable number of labels for each student in each lesson. These observations enabled us to calculate the proportion of time each student spent gaming in each lesson.

2. The Cognitive Tutor Lesson Variation Space (CTLVS1)

Next we developed an enumeration of the ways that Cognitive Tutor lessons can differ from one another. This enumeration, in its current form, is called the Cognitive Tutor Lesson Variation Space version 1.0 (CTLVS1). The CTLVS1 was developed by a six member design team with diverse expertise, including three Cognitive Tutor designers (with expertise in cognitive psychology and artificial intelligence), a researcher specializing in the study of gaming the system, a mathematics teacher with several years of experience using Cognitive Tutors in class, and a designer of non-computerized curricula who had not previously used a Cognitive Tutor.

The first step of the design process was for each participant who had previously used a Cognitive Tutor to separately – from memory – write a list of the features a tutor lesson could have that would differentiate it from another tutor. Afterwards, four of the participants separately went through each lesson in Cognitive Tutor Algebra, and wrote down any new features that came to mind. One participant conducted a literature review on papers discussing differences in intelligent tutors and similar forms of computer-aided instruction, and changes in tutor design over time, to generate further features. In total, the first step of the design process generated a list with 569 features.

The second step was to develop a list of criteria for features that would be worth coding, to narrow this list down to a more tractable size. The criteria settled upon consisted of being understandable by multiple team members, whether a feature could probably be reliably coded (or automatically generated via data mining), whether a feature split the lesson space into two sizable groups (as opposed to distinguishing only 1 or 2 lessons) (this criterion did not apply to software bugs, which are expected to be rare in a widely used commercial product such as Cognitive Tutor Algebra, but might still be worth associating with gaming), and whether the group thought a feature had a reasonable chance of being associated with gaming behavior.

Each feature was evaluated according to these criteria by three members of the design team. 113 features were judged by the group to satisfy the set of criteria to a sufficient degree to be worth including in the CTLVS1. Then (a non-identical) three members of the design team proceeded to code the set of features for each of the 22

lessons in the Algebra tutor for which we had data on gaming frequency. During the coding process, 34 features were found to be intractable, either because they took too much time to code, or because the coders felt uncertain about codes in multiple cases. This left a set of 79 quantitative and binary features (which were treated as quantitative during analysis). Of the 79 features, 10 were coded via data mining; the other 69 were coded by hand. Inter-rater reliability checks were not conducted, owing to the hypothesis-generating nature of this study. The 79 features within the CTLVS1 which were coded are shown in Table 1. Features are grouped into approximate categories within Table 1.

The CTLVS1 was explicitly designed for use in comparing Cognitive Tutors. However, many of the features would also be relevant in other intelligent tutors, and in computer-aided instruction more broadly. Hence the effort to design the CTLVS1 has the potential to be useful not just in enabling the analysis presented here, but also in the broader study of how users respond to differences in computer-aided instruction.

Table 1. The 79 features of the Cognitive Tutor Lesson Variation Space (CTLVS1). Features distilled through data mining (as opposed to hand-coding) are marked with *.

Difficulty, Complexity of Material, and Time-Consumingness	
1*. Avg. % error	2. Lesson consists solely of review of material encountered in previous lessons
3*. Avg. probability that student will learn a skill at each opportunity to practice skill [cf. 10]	4*. Avg. initial probability that student will know a skill when starting tutor [cf. 10]
5. Avg. # of "distractor" values per problem	6. % of problems where "distractor" values given
7. Max number of mathematical operators needed to give correct answer on any step in lesson	8. Maximum number of mathematical operators mentioned in hint on any step in lesson
9. Intermediate calculations must be done outside of software (mentally or on paper) for some problem steps (ever occurs)	10. % of hints that discuss intermediate calculations that must be done outside of software
11*. Total number of skills in lesson	12*. Avg. time per problem step
13. % of problem statements that incorporate multiple representations (ex: diagram and text)	14. % of problem statements that use same numeric value for two constructs
15. Avg. number of distinct/separable questions or problem-solving tasks per problem	16. Maximum number of distinct/separable questions or problem-solving tasks in any problem
17. Avg. # of numbers manipulated per step	18. Avg. # of times each skill repeated per problem
19*. Number of problems in lesson	20*. Avg. time spent in lesson
21. Avg. number of problem steps per problem	22. Minimum number of answers or interface actions required to complete problem
Quality of Help Features	
23*. Avg. amount that reading on-demand hints improves performance on future opportunities to use skill [cf. 9]	24*. Avg. Flesch-Kincaid Grade Reading Level [12] of hints, a measure of how difficult the hints were to read (in terms of vocabulary and grammar)
25. % of hints using inductive support, going from example to abstract concept/principle	26. % of hints that explicitly explain concepts or principles underlying current problem-solving step
27. % of hints that explicitly refer to abstract principles	28. On average, # of hints must student request before concrete features of problems are discussed
29. Avg. number of hint messages per hint sequence that orient student to math sub-goal	30. % of hints that explicitly refer to scenario content (instead of solely math constructs)
31. % of hint sequences that use terminology specific to this software	32. % of hint messages which refer solely to interface features
33. % hint messages that teacher can't understand	34. % of hint messages with complex noun phrases
35. % of skills where the only hint message explicitly tells student what to do	
Usability	

36. First problem step in first problem of lesson is either clearly indicated, or follows established convention (such as top-left cell in worksheet)	37. % of steps where student must change a value in a cell that was previously labeled as correct (example: estimation lessons)
38. After student completes step, system indicates where in interface next action should occur	39. % of steps where it is necessary to request hint to figure out what to do next
40. Not immediately apparent what icons in toolbar mean	41. Screen cluttered with interface widgets; difficult to determine where to enter answers
42. Problem-solving task is not immediately clear	43. Format of answer changes between problem steps without clear indication
44. If student has skipped step, and asks for hint, hints refer to skipped step without explicitly highlighting in interface (ever seen)	45. If student has skipped step, and asks for hint, skipped step is explicitly highlighted in interface (ever seen)
Relevance and Interestingness	
46. % of problems which appear to use real data	47. % of problem statements with story content
48. % of problem statements with scenarios relevant to the "World of Work" [cf. 11]	49. % of problem statements with scenarios relevant to students' current daily life
50. % of problem statements which involve fantasy (example: being a rock star)	51. % of problem statements which involve concrete details unfamiliar students (example: dog sleds)
52. % of problem statements which involve concrete people/places/things	53. % of problem statements with text not directly related to problem-solving task
54. Avg. number of person proper names in problem statements	
Aspects of "buggy" messages notifying student why action was incorrect	
55. % of buggy messages that indicate concept student demonstrated misconception in	56. % of buggy messages that indicate how student's action was result of procedural error
57. % of buggy messages that refer solely to interface action	58. Buggy messages given by icon, which can be hovered over to receive buggy message
Design Choices Which Make It Easier to Game the System	
59. % of multiple-choice steps	60. Avg. number of choices in multiple-choice
61. % of hint sequences with final "bottom-out" hint that explicitly tells student what to enter	62. % of hint sequences with final hint that explicitly tells student what the answer is, but not what/how to enter it in the tutor software
63. Hint gives directional feedback (example: "try a larger number") (ever seen/boolean)	64. Avg. number of feasible answers for each problem step
Meta-Cognition and Complex Conceptual Thinking (or features that make them easy to avoid)	
65. Student is prompted to give self-explanations	66. Hints ever give explicit metacognitive advice
67. % of problem statements that use common word to indicate mathematical operation to use (example: "increase")	68. % of problem statements that indicate math operation with uncommon terminology ("pounds below normal" for subtraction)
69. % of problem statements that explicitly tell student which math operation to use ("add")	
Software Bugs/Implementation Flaws (generally rare)	
70. % of problems where grammatical error is found in problem statement	71. Reference in problem statement to interface component that does not exist (ever occurs)
72. Student can advance to new problem despite still visible errors	73. Hint recommends student do something which is incorrect or non-optimal (ever occurs)
74. % of problem steps where hints are unavailable	
Miscellaneous	
75. Hint requests that student perform some action	76*. Avg. length of text in popup widgets
77. Value of answer is very large (over four significant digits) (ever seen)	78. % of problem statements which include question or imperative
79. Student selects action from menu, tutor software performs action (as opposed to typing in answers, or direct manipulation)	

3. Analysis Method and Results

The goal of our analyses was to determine how well each difference in lesson features predicts how much students will game in a specific lesson. To this end, we combined the labels of the CTLVS1 features for each of the 22 Cognitive Tutor Algebra lessons studied, with the assessments of how often each of the 58 students in the data set gamed the system in each of the 22 lessons.

Our first step in conducting the analysis was to determine if the 79 features of the CTLVS1 grouped into a smaller set of categories. One option would have been to have used the categories of features within Table 1 – however, it is not clear that the features in each category would be inter-correlated, as they cover different aspects of large constructs, and the assignments to categories are in any event approximate. Instead, we empirically grouped the 79 features of the CTLVS1 into 6 factors, using Principal Component Analysis (PCA) (a similar pattern was observed with other numbers of factors), a common method for distilling inter-item structure.

We then analyzed whether the correlation between any of these 6 factors and the frequency of gaming the system was significant. Of the 6 factors, one was statistically significantly associated with the choice to game the system, $r^2 = 0.29$ (e.g. accounting for 29% of the variance in gaming), $F(1,19)= 7.84$, $p=0.01$. The factor loaded strongly on eight features associated with more gaming:

- 14: The same number being used for multiple constructs
- 23-inverse: Reading hints does not positively influence performance on future opportunities to use skill
- 27: Proportion of hints in each hint sequence that refer to abstract principles
- 40: Not immediately apparent what icons in toolbar mean
- 53-inverse: Lack of text in problem statements not directly related to the problem-solving task, generally there to increase interest
- 63-inverse: Hints do not give directional feedback ("try a larger number")
- 71-inverse: Lack of implementation flaw in hint message. Flaw is a reference to a non-existent interface component
- 75: Hint requests that student perform some action

In general, several of the features in this factor appear to correspond to a lack of understandability in the presentation of the content or task (23-inverse, 40, 63-inverse), as well as abstractness (27) and ambiguity (14). Curiously, feature 71-inverse (the lack of a specific type of implementation flaw in hint messages, which would make things unclear) appears to point in the opposite direction – however, this implementation flaw was only common in a single rarely gamed lesson, and thus may be a statistical artifact.

Feature 53-inverse appears to represent a different construct – the attempt to increase interestingness. The fact that feature 53 was associated with less gaming whereas more specific interest-increasing features (features 46-52) were not significantly related may suggest that it is less important exactly how a problem scenario attempts to increase interest, than it is important that the problem scenario has some content in it that is not strictly mathematical.

Taken individually, two of the constructs in this factor were significantly (or marginally significantly) associated with gaming. Feature 53-inverse (text in the problem statement not directly related to the problem-solving task) was associated with significantly less gaming, $r^2 = 0.19$, $F(1,19) = 4.59$, $p = 0.04$. Feature 40 (when it is not

immediately apparent what icons in toolbar mean) was marginally significantly associated with more gaming, $r^2 = 0.15$, $F(1, 19)=3.52$, $p=0.08$. The fact that other top features in the factor were not independently associated with gaming, while the factor as a whole was fairly strongly associated with gaming, suggests that gaming occurs primarily when more than one of these features are present. Only one other taxonomy feature was significantly associated with gaming: Feature 36, where the location of the first problem step is not directly indicated and does not follow standard conventions (such as being the top-left cell of a worksheet), $r^2 = 0.20$, $F(1,19)=4.97$, $p=0.04$. This feature, like many of those in the gaming-related factor, implies an unclear or confusing lesson.

Further analysis indicated that Feature 53-inverse can be meaningfully split into two sub-features: Whether there was any text at all in the problem statement that was not directly related to the problem-solving task (binary feature), and how much text there was, among problems that did have at least some text (numerical feature). Curiously, having less text (if there was any text at all) was associated with more gaming, but problems with no text at all appeared to have less gaming than would be predicted solely from their degree of text. One possibility is that other features of the problems with no text at all led to the lower gaming (in particular, these lessons focused on equation-solving, a skill that is different in kind from more complex problem-solving). A model that splits out these sub-features (in addition to the factor and Feature 36) is significantly better at predicting gaming than a model which does not split these sub-features, $F(1,19)=6.14$, $p=0.02$.

The full model, including the factor and all other significant features, explains 56% of the variance in gaming ($r^2 = 0.56$), *over five times* the degree of variance in gaming explained by any prior study predicting gaming with student individual differences.

4. Discussion and Conclusions

In this paper, we have studied which specific attributes of Cognitive Tutors lead to differences in how much students game the system, by developing a taxonomy of ways that tutors can differ from each other, labeling a set of tutor lessons with reference to this set of features, and identifying which features are associated with gaming. In past studies predicting gaming with student individual differences, the strongest relationships between student individual differences and gaming were around $r^2 = 0.09$. In this study, one difference between lessons achieved an r^2 of 0.20, by itself explaining about twice as much of the variance in gaming as any previously studied difference between students, and a full model based on the taxonomy achieved an r^2 of 0.56.

The results suggest that gaming the system appears to be significantly more frequent in lessons that are abstract, ambiguous, and have unclear presentation of the content or task, a finding that coheres with the previous finding that students tend to display confusion shortly before gaming [15]. The finding that less gaming occurs in lessons with non-task related text in the problem statement (included to increase interest) coheres with the previous finding that boredom also precedes gaming [15].

Beyond this confirmation of earlier results, these findings offer concrete prescriptions for how tutor lessons can be re-designed to reduce gaming, suggesting that we may be able to reduce gaming in intelligent tutoring systems substantially through the following feasible steps: First, re-designing confusing toolbars to make them easier to understand, perhaps with pop-up messages for each toolbar item

explaining how it is used. Second, re-designing specific hints that do not seem to effectively improve student future performance [cf. 9]. Third, replacing textual references to abstract principles in hints with other ways of communicating abstract principles, such as diagrams or interactive examples. Fourth, adding more interest-increasing text to problem scenarios (when they have text to begin with).

These prescriptions can be tested in future research, by taking a frequently-gamed lesson with these features, and re-designing the lesson as these findings suggest. If the resultant lesson is gamed significantly less, and students have better learning, we will have additional confirmation of the hypotheses generated here for why students game, and an actionable plan for how to reduce gaming in interactive learning software.

Beyond this, we believe that the methods used in this paper point to a new way that student-tutor interactions can be studied in a deep fashion. The creation of taxonomies such as the CTLVS1 will enable an increasing number of data mining analyses about how differences in educational software concretely influence student behavior. In turn, these analyses will enable the development of more precise and validated guidelines for the design of educational software.

References

[1] 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. *Proceedings of the 13th International Conference on Artificial Intelligence in Education (AIED-2007)*, 195-202.

[2] Arroyo, I., & Woolf, B. Inferring learning and attitudes from a Bayesian network of log file data. *Proceedings of the 12th International Conference on Artificial Intelligence in Education* (2005), 33-40.

[3] 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. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems* (2006), 392-401.

[4] 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". *Proc. ACM CHI 2004*, 383-390.

[5] Baker, R.S.J.d. Is Gaming the System State-or-Trait? Educational Data Mining Through the Multi-Contextual Application of a Validated Behavioral Model. *Proc. of the Workshop on Data Mining for User Modeling at the 11th International Conference on User Modeling 2007*, 76-80.

[6] Baker, R.S.J.d., Corbett, A.T., Wagner, A.Z. Human Classification of Low-Fidelity Replays of Student Actions. *Proc. Edu. Data Mining Workshop, Int'l Conf. on Intelligent Tutoring Systems* (2006), 29-36.

[7] 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* ***19*****(2)** (2008), 185-224.

[8] Beal, C.R., Qu, L., Lee, H. Mathematics motivation and achievement as predictors of high school students' guessing and help-seeking with instructional software. *J. Comp. Assisted Learning,* in press.

[9] Beck, J.E. Using Learning Decomposition to Analyze Student Fluency Development. *Proc. Edu. Data Mining Workshop, 8th Int'l Conf. on Intelligent Tutoring Systems* (2006), 21-28.

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

[11] Csikszentmihalyi, M., Schneider, B. *Becoming Adult: How Teenagers Prepare for the World of Work.* New York: Basic Books, 2000.

[12] Kincaid, J. P., Fishburne, R. P., Rogers, R. L., Chissom, B. S. Derivation of new readability formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy enlisted personnel. Research Branch Report 8-75, Naval Technical Training, 1975.

[13] Koedinger, K., Cunningham, K., Skogsholm A., Leber, B. An open repository and analysis tools for fine-grained, longitudinal learner data. *Proc. 1st Int'l Conf. on Edu. Data Mining* (2008), 157-166.

[14] Murray, R.C., & vanLehn, K. Effects of dissuading unnecessary help requests while providing proactive help. *Proc. of the 12th Int'l Conf. on Artificial Intelligence in Education* (2005), 887-889.

[15] 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 the 13th International Conference on Artificial Intelligence in Education* (2007), 145-152.

[16] Roll, I., Aleven, V., McLaren, B. M., Koedinger, K. R. Can help seeking be tutored? Searching for the secret sauce of metacognitive tutoring. *Proceedings of the 13th International Conference on Artificial Intelligence in Education* (2007).

[17] Walonoski, J.A., & Heffernan, N.T. Prevention of off-task gaming behavior in intelligent tutoring systems. *Proc. of the 8th International Conference on Intelligent Tutoring Systems* (2006), 722-724.