Reengineering the Feature Distillation Process: A Case Study in the Detection of Gaming the System

Luc Paquette, Adriana M.J.A. de Carvalho, Ryan S. Baker, & Jaclyn Ocumpaugh
Teachers College Columbia University
525 West 120th Street
New York, NY, 10027
paquette@tc.columbia.edu

ABSTRACT

As education technology matures, researches debate whether data mining (EDM) or knowledge engineering (KE) paradigms are best for modeling complex learning constructs. A hybrid paradigm may capture strengths from both approaches. In particular, recent work has argued that successful data mining depends on thoughtful feature engineering. In this paper, we explore the use of cognitive modeling (a form of knowledge engineering) to enhance the feature engineering process for detectors of gaming the system, one of the most studied complex constructs in EDM. Using this construct enables us to measure the extent to which our techniques improve performance over previous models.

Keywords

Gaming the system, Cognitive Tutor, feature engineering, cognitive modeling, cognitive task analysis

1. INTRODUCTION

Over the last ten years, researchers interested in student disengagement have sought to improve the detection of gaming the system, behavior where students attempt to solve problems in an educational environment by exploiting properties of the system [2]. Within intelligent tutors, gaming the system manifests in several ways, including help abuse (e.g. [1], [12], [17]) and systematic guessing (e.g. [12], [17]). However, the construct appears to be quite complex, and while human coders are capable of achieving good inter-rater reliability for this construct [2], its complexity is still a challenge for the modeling community.

Gaming the system has now been modeled in a variety of systems using techniques from both Educational Data Mining (EDM) and Knowledge Engineering (KE). Within an EDM approach, classification algorithms are used to match training labels generated from *in situ* field observations (as in [17]) or from text replays (e.g. [2], [3] and [4]). These models have been effective at predicting gaming, but critics of EDM techniques argue that the resultant models are difficult to interpret.

KE models of gaming offer greater interpretability, but may oversimplify a construct that can manifest in many different ways. Often KE models focus only on 1-2 patterns of gaming, (e.g. quick incorrect answers or specific types of help abuse in [12], [17]), and it is reasonable to question whether such a complex and ill-defined construct can be fully described by 2-3 simple rules. In particular, simple rules may indicate gaming when a student skips to bottom-out hints to obtain answers (a pattern typical of gaming), but then pauses to self-explain, a behavior associated with positive learning outcomes [16].

In this study, we leverage Cognitive Task Analysis (CTA) [5], a form of KE, to produce a better EDM model. In line with results suggesting that attention to feature construct validity improves model goodness [15], we enhance construct validity by constructing features based on the explicit patterns articulated by expert human judges for how they recognize gaming [13]. We find that this method generates features that better reflect the meaningful units of student behavior that trigger experts to recognize gaming; using these features within an EDM process leads to better goodness than a model developed using cognitive task analysis alone.

2. COGNITIVE MODELING OF TEXT REPLAY CODING

Many EDM studies of gaming that have leveraged human judgments have relied upon text replays ([3], [4]), a sequence (clip) of student actions displayed in textual form, used to reliably and rapidly label systematic patterns of student behavior. Each replay contains time-stamped information about the context the student is interacting with (elements of the learning environment, including the relevant skills being tested), the input entered by the student, and the system's assessment of that input (right, wrong, a "bug" or common misconception, or a help request). A trained human coder labels each text replay as "gaming" or "not gaming".

In the CTA presented in [13], researchers interviewed and observed a gaming expert who had coded over 20,000 text replays from Cognitive Tutor Algebra, eliciting information about which cues were meaningful during that process. [13]'s CTA showed that expert coding involved two main processes: interpreting the student's actions and using these interpretations to identify patterns indicative of gaming. In particular, CTA identified 19 different constituents, or units of behavior, used by the expert. Analysis shows that the expert relied heavily on pauses to assess students' reflection and engagement, but gaming labels were also dependent upon contextualized information about the student input (e.g., was the student entering several similar answers in a row). Table 1 shows a partial list of these constituents.

Further analysis [13] found that no constituent is independently sufficient for identifying gaming, but that certain combinations of constituents are. Expert interviews identified 13 substantive patterns of the 19 constituents, which we refer to as *pattern features*. In this paper, we build on this work, using the constituents of [13]'s CTA to generate new pattern features and then applying EDM techniques to improve model performance.

Table 1. Some pattern constituents indicative of gaming

Constituent Description		Interpretive Label
C1	Pause ≤ 5 seconds before a help request	*[did not think before help request]*
C2	Pause ≥ 4 and ≤ 8 seconds per help message after a help request	*[scanning help messages]*
C3	Pause ≤ 3 seconds per help message after a help request	*[searching for bottom out hint]*
C4	Pause ≤ 5 seconds before a step attempt	*[guess]*
C5	Pause ≤ 8 seconds after a bug	*[did not read error message]*
C6	Answer was the same as the previous action, but in a different context	*[same answer/diff. context]*
C7	Answer was similar to the previous one (Levenshtein [16] distance of 1 or 2)	*[similar answer]*
C8	Context of the current action is not the same as the context for the previous	*[switched context before right]*
C9	Context for the current action is the same as the context for the previous	*[same context]*
C10	Answer or context is not the same as the previous action	*[diff. answer AND/OR diff. context]*

3. METHODS

3.1 Data

This study relies on data from Cognitive Tutor Algebra that have been used to study three gaming models ([3], [12], [13]), the Pittsburgh Science of Learning Center DataShop "Algebra I 2005-2006 (Hampton only)" dataset [8], which contains data from 59 students over the course of an entire school year. For [3], the data from 12 different lessons were segmented into clips of at least five actions and 20 seconds in length, within a single problem. A total of 10,397 text replays were presented to the expert, who labeled 708 (6.8%) as gaming [3].

[13] divided these text replay clips into subsets so that 75% of the clips from each category (531 gaming and 7,267 not gaming) are randomly assigned to a training set. The remaining 25% (177 gaming and 2,422 not gaming) were held-out for testing to ensure against overfitting. In this study, we use the same division during feature distillation, but final models are trained using standard cross-validation techniques.

3.2 Feature Distillation

The 19 constituents identified through [13]'s cognitive task analysis were used as features for the detectors built in this study. Constituent labels were applied to each clip, and the number of times each constituent appeared was computed.

Whereas the cognitive modeling approach in [13] attempted to replicate the expert's decision process, this paper's hybrid model searches for pattern features beyond those the expert directly articulated. This enables us to test a broader range of patterns on the large number of clips coded by the expert coder.

In order to generate new patterns, each clip from the training set was tagged with the 19 constituents identified in [13]. Constituent labeling involved a multistep process. First, student actions were given the following 4 labels: *help*, *attempt* (an attempted answer regardless of its correctness), *incorrect* (bug or wrong attempt) and *bug*. Note that the range in specificity here allows more than one action label to be applied in some cases. Next, 15 2-action and 57 3-action sequences were created from these action labels. For example, the 2-action sequences included "help → attempt" and "help → incorrect," and 3-action sequences included "incorrect → help → attempt." Sequences of 2 consecutive help requests were not generated since these are collapsed in the log files. Next, these sequences were tagged with constituent labels. In order to reduce the number of possible combinations, constituents that were associated with "not gaming" in the CTA were excluded from this process. These labels were then used to generate patterns consisting of 0-2 constituents. Impossible combinations of constituents were excluded (e.g., a help request could not be tagged with both [*scanning help message*] (C2) and [*searching for bottom out hints*] (C3)), producing 496,944 possible pattern features.

As the feature set was now enormous (increasing the potential for over-fitting), 2 steps were used to reduce the number of features tested in our final model. First, Cohen's Kappa [6] was used to evaluate how individual pattern features predicted gaming. Pattern features with Kappa < 0.05 (Kappa of 0 indicates chance) were eliminated first, reducing the number of possible features from 496,944 to 29,294. Next, a modified forward selection process was applied to the remaining patterns.

Although Kappa is a popular indicator of performance, it is relatively poor at eliminating patterns that identify many true positives at the cost of also identifying a large numbers of false positives. Since a combination of more specific sub-patterns might detect just as many true positives while detecting fewer false positives, a combination of more specific pattern features could achieve a better performance, even though the single overly general pattern would be selected first by a forward selection process based solely on Kappa. In order to prevent high rates of false positives, our forward selection process gave more weight to pattern features with a higher ratio of true positives (TP) to false positives (FP), a metric similar to precision. For the first iterations of our forward selection process, only pattern features with a TP to FP ratio ≥ to 1 were considered. This threshold was then lowered in increments of .05 each time Kappa could no longer be improved at the current threshold. This process repeated until the threshold became 0 and Kappa did not improve.

The ratio of TP to FP was used during forward selection instead of precision to reduce over-fitting to the training set. Amongst the generated pattern features, many detect a small number of TP while not capturing any FP. Those pattern features are likely to be overly specific to the training set. For such patterns, the value for the precision metric will be 1, the highest possible value, whereas the ratio of TP to FP is undefined (and are treated as 0 in our approach). As such, when executing forward selection using precision, those overly specific patterns will be added early to the set of best patterns, over-fitting to the training set. By contrast, those patterns will only be considered as possible best patterns when using TP to FP ratio if they still contribute to the overall performance at the end of the forward selection process.

Performance was evaluated on both the training and the test set to ensure that our forward selection algorithm did not overfit to the training data. This process was executed on the training set, resulting in the selection of 60 pattern features.

In addition to constituent and pattern features, 6 features, which we term "count features," were also considered. For these, we counted the number of actions of specific types during the clip, including (1) help, (2) attempts, (3) right answers, (4) incorrect answers (whether just wrong or a bug), (5) wrong answers (incorrect but not a bug), and (6) bugs. Combined with the other 2 feature types, this resulted in 85 features that were considered during the construction of *CognitiveHybrid-PF*, our first hybrid detector.

3.3 Validation and Performance

Detectors of gaming the system were constructed in RapidMiner 5.3 [11], using J48, JRip, Step Regression and Naïve Bayes, four algorithms that have been successful for past educational data mining problems. Performance was assessed using two metrics: Cohen's Kappa and A' [7]. A' is the probability that given a pair of two clips, one coded as gaming the system and the other coded as non-gaming, the model can accurately detect which clip was coded as gaming. A' is equivalent to the area under the ROC curve in signal detection theory and the Wilcoxon statistic [7]. A detector with an A' of 0.5 performs at chance, and a detector with an A' of 1.0 performs perfectly. A' was computed at the clip level, using the code at http://www.columbia.edu/~rsb2162/edmtools.html.

Detector performance during RapidMiner's forward selection was evaluated using a 6-fold student-level cross-validation. By cross-validating at the student level, we increase the confidence that our detectors will generalize to new students.

4. Results

The detectors were refined in three stages.

4.1 *CognitiveHybrid-PF*: Pattern Feature Detector

Our first detector was built using all 85 features. The Naïve Bayes algorithm performed best under 6-fold student-level cross-validation. (Kappa = 0.477 and A' = 0.770), accurately diagnosing 411 (58.05%) of the gaming clips and misdiagnosing only 495 (5.11%) of the non-gaming clips. The resulting model (Table 2) contains 22 of 85 potential features: 20 pattern features, 1 constituent-based feature (F21), and 1 count feature (F22). Except for F22, each was associated with a higher probability of gaming by the Naïve Bayes detector.

A closer inspection of the model improves our understanding of the actions and constituents that typify gaming in Cognitive Tutor Algebra. Only 3 pattern features (F5, F8, F12) selected in this model contain help constituents (C1 and C3). Instead, the predominant label was *incorrect* (both bugs and other wrong answers). This action appeared in 19/20 pattern features, omitted only from F7, where further scrutiny shows that the more specific *bug* label was a component of this and 7 other pattern features. This suggests that incorrect answers typify gaming, but a contrasting result also emerges. Right answers were possible in 13 of 20 pattern features, perhaps because a student who is gaming the system might get the correct answer by systematically guessing.

Overall, *CognitiveHybrid-PF's* feature selection suggests gaming behaviors in this corpus are typified by fast, systematic guessing patterns (e.g., providing similar answers or the same answers in different contexts). The effect of context changes appears to be nuanced but highly predictive when combined with other factors. A student who repeatedly enters the same answer in different contexts is not engaged in learning, but neither is a student who persists within one context after multiple incorrect steps.

4.2 *CognitiveHybrid-C*: Constituent Detector

Although *CognitiveHybrid-PF* shows substantial improvements over previous models in terms of cross-validated Kappa, room for improvement remains, especially for A'. Within *CognitiveHybrid-PF*, A' may have been reduced by the binary way the pattern features were used, resulting in high confidences for clips matching one or more pattern features, but confidences approaching 0 for all other clips. To address this issue, we construct

Table 2. Features utilized by *CognitiveHybrid-PF*.

Selected features	
F1	**incorrect** → [*same answer/diff. context*] & **incorrect**
F2	[*diff. answer AND/OR diff. contex* t] & **incorrect** → [*similar answer*] & **incorrect** → [*similar answer*] & **incorrect**
F3	**bug** & [*did not read error message*] → [*similar answer*] & **incorrect** → [*diff. answer AND/OR diff. context*] & **attempt**
F4	**incorrect** → [*same answer/diff. context*] & **attempt** → **bug**
F5	[*similar answer*] & **incorrect** → [*guess*] & [*similar answer*] & **attempt** → [*did not think before help request*] & [*same context*] & **help**
F6	**incorrect** → [*guess*] & [*similar answer*] & **attempt** → [*switched context before right*] & **incorrect**
F7	**bug** → [*guess*] & diff. answer AND/OR diff. context] & **bug** → [*guess*] & [*diff. answer AND/OR diff. context*] & **attempt**
F8	[*did not think before help request*] & [*same context*] & **help** → **attempt** → [*guess*] & [*similar answer*] & **incorrect**
F9	**bug** → [*similar answer*] & **incorrect** → [*diff. answer AND/OR diff. context]* & **bug**
F10	[*guess*] & [*same context*] & **attempt** → [*same context*] & **incorrect** → [*guess*] & [*similar answer*] & **incorrect**
F11	[*guess*] & [*same context*] & **incorrect** → [*diff. answer AND/OR diff. context*] & **attempt** → [*switched context before right*] & **incorrect**
F12	[*guess*] & [*similar answer*] & **incorrect** → [*diff. answer AND/OR diff. context*] & **incorrect** → **help** & [*searching for bottom-out hint*]
F13	**incorrect** → [*similar answer*] & **bug** → [*same answer/diff. context*] & **attempt**
F14	[*guess*] & [*diff. answer AND/OR diff. context*] & **incorrect** → [*guess*] & **bug** → [*guess*] & [*diff. answer AND/OR diff. context*] & **attempt**
F15	[*same context*] & **bug** & [*did not read error message*] → [diff. answer AND/OR diff. context] & **attempt** → [*guess*] & **incorrect**
F16	[*guess*] & [*same answer/diff. context*] & **attempt** → **incorrect** → [*diff. answer AND/OR diff. context*] & **incorrect**
F17	**incorrect** → [*guess*] & [*diff. answer AND/OR diff. context*] & **bug** → [*diff. answer AND/OR diff. context*] & **incorrect**
F18	[*guess*] & [*same context*] & **incorrect** → [*diff. answer AND/OR diff. context*] & **incorrect** → [*similar answer*] & **incorrect**
F19	[*similar answer*] & **incorrect** → [*same context*] & **incorrect** → [*similar answer*] & **incorrect**
F20	[*similar answer*] & **incorrect** → [*guess*] & [*similar answer*] & **incorrect** → [*similar answer*] & **attempt**
F21	number of times that [*switched context before right*] occured in the clip
F22	number of **right answers** in the clip

CognitiveHybrid-C, which relies only on constituent and count features. As with *CognitiveHybrid-PF*, Naïve Bayes was selected as the best algorithm when performance was assessed using 6-fold student-level cross validation. The exclusion of pattern features improved A' (0.875) but also increases the false positive rate, lowering Kappa (0.332). The model accurately diagnosed 323 (45.62%) gaming clips but misdiagnosed 657 (6.78%) non-gaming clips. Compared to *CognitiveHybrid-PF*, *CognitiveHybrid-C* is more parsimonious, requiring only 2 constituent features ([same answer/diff. context] and [thought about error]) and 4 count features (wrong, bug, incorrect, and right). Except for (count of right), all were associated with higher probabilities of gaming.

4.3 *CognitiveHybrid-E*: Ensemble Detector

Both *CognitiveHybrid-C and CognitiveHybrid-PF* have strengths, but neither is ideal. *CognitiveHybrid-E* (our ensemble detector) leverages the better prediction confidences (A') of *C* and the better classifications (Kappa) of *PF* by ensembling the two. This is done by averaging the two models' confidences together, and setting a threshold of 0.5. *CognitiveHybrid-E*, when student-level cross-validated, achieves good Kappa (0.457) and A' (0.901), accurately diagnosing 392 (55.37%) gaming clips and misdiagnosing only 476 (4.91%) not-gaming clips.

CognitiveHybrid-E's performance (Kappa = 0.457, A' = 0.901) is better than previous detectors trained on the same data. Neither [3]'s decision tree detector (Kappa = 0.40) nor their latent response model (Kappa = 0.04) is cross-validated [3] and (unpublished) cross-validation drops the decision tree detector's performance to Kappa = 0.24. [13]'s cognitive model performed well on training data (Kappa = 0.430), but performance dropped when applied to a held-out test set (Kappa = 0.330). *CognitiveHybrid-E* also compares favorably to other published gaming detectors: [4], conducted in SQL-Tutor, reported student-level cross-validated Kappa = 0.36, A' = 0.770. In ASSISTments [14], a model of gaming achieved Kappa = 0.370 and A' = 0.802; an earlier model in ASSISTments [17] achieved Kappa = 0.181.

5. CONCLUSIONS AND DISCUSSION

In this study, we provide enhanced, automated models of gaming-the-system for Cognitive Tutor Algebra, improving model performance for a construct already well established in the literature ([3], [13]). Improvements were driven by a hybrid approach that leverages both KE and EDM techniques, using cognitive modeling of human experts during feature distillation and then applying EDM practices to combine these operators to predict gaming.

These results have implications for debates between KE and EDM approaches. They suggest that EDM researchers could substantially improve their feature engineering by employing KE techniques during feature distillation. At the same time, they also attest to limitations in relying solely on human experts to define the constructs in automated detectors. There are many constructs that humans can easily recognize but are still difficult to define. The detailed interview method used in [13] and built on here foregrounds the value of expert evaluations. By considering hundreds of thousands of possible patterns, EDM methods can improve performance. For modeling complex constructs, the combination of KE and EDM can be stronger than either method alone.

6. ACKNOWLEDGMENTS

This research was supported by a FRQNT (Fonds de recherche du Québec - Nature et technologies) post-doctoral fellowship and by National Science Foundation grant #SBE-0836012.

7. REFERENCES

[1] Aleven, V., McLaren, B. M., Roll, I., Koedinger, K. R. 2004. Towards Tutoring Help Seeking: Applying Cognitive Modeling to Meta-Cognitive Skills. *Proc of ITS 2004*, 227-239.

[2] Baker, R. S. J. d., Corbett, A. T., Wagner, A. Z. 2006. Human Classification of Low-Fidelity Replays of Student Actions. *Proc of EDM Workshop at ITS 2006*, 29-36.

[3] Baker, R. S. J. d., de Carvalho, A. M. J. A. 2008. Labeling Student Behavior Faster and More Precisely with Text Replays. *Proc of EDM 2008*, 38-47.

[4] Baker, R. S. J. d., Mitrovic, A., Mathews, M. 2010. Detecting Gaming the System in Constraint-Based Tutors. *Proc of UMAP 2010*, 267-278.

[5] Clark, R. E., Feldon, D., van Merriënboer, J., Yates, K., Early, S. 2008. Cognitive Task Analysis. In *Handbook of Research on Educational Communications and Technology (3rd ed.)*, 575-593.

[6] Cohen, J. 1960. A Coefficient of Agreement for Nominal Scales. *Educational & Psych Measurement*, 20,11, 37-46.

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

[8] Koedinger, K. R., Baker, R., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J. 2010. *A Data Repository for the Community: The PLSC DataShop*. CRC Press, Boca Raton.

[9] Koedinger, K. R., Corbett, A. T. 2006. Cognitive Tutors: Technology Bringing Learning Sciences to the Classroom. *The Cambridge Handbook of the Learning Sciences*, 61-77.

[10] Levenshtein, A. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10, 8, 707-710.

[11] Mierswa, I., Scholz, M., Klinkenberg, R., Wurst, M., Euler, T. 2006. Yale: Rapid Prototyping for Complex Data Mining Tasks. Proc of KDD 2006, 935-940.

[12] Muldner, K., Burleson, W., Van de Sande, B., VanLehn, K. 2011. An Analysis of Students' Gaming Behaviors in an Intelligent Tutoring System: Predictors and Impact. *User Modeling and User Adapted Interaction*, 21, 99-135.

[13] Paquette, L., de Carvalho, A. M. J. A., Baker, R. S. *accepted*. Towards Understanding Expert Coding of Student Disengagement in Online Learning. *Proc. Cog Science Society*.

[14] Pardos, Z.A., Baker, R.S., San Pedro, M.O.C.Z., Gowda, S.M., Gowda, S.M. *in press*. Affective States and State Tests: Investigating How Affect and Engagement During the School Year Predict End of Year Learning Outcomes. To appear in *Journal of Learning Analytics*.

[15] Sao Pedro, M., Baker, R., Gobert, J. 2012. Improving Construct Validity Yields Better Models of Systematic Inquiry, Even with Less Information. *Proc of UMAP 2012*, 249-260.

[16] Shih, B., Koedinger, K. R., Scheines, R. 2008. A Response Model for Bottom-Out Hints as Worked Examples. *Proc of EDM 2008*, 117-126.

[17] Walonoski, J. A., Heffernan, N.T. 2006. Detection and Analysis of Off-Task Gaming Behavior in Intelligent Tutoring Systems. *Proc of ITS 2006*, 382-391.