Comparing machine learning to knowledge engineering for student behavior modeling: a case study in gaming the system

Luc Paquette & Ryan S. Baker

INTERACTIVE LEARNING ENVIRONMENTS
2019, VOL. 27, NOS. 5–6, 585–597
https://doi.org/10.1080/10494820.2019.1610450

Comparing machine learning to knowledge engineering for student behavior modeling: a case study in gaming the system

Luc Paquette[a] and Ryan S. Baker[b]

[a]Department of Curriculum & Instruction, University of Illinois at Urbana-Champaign, Champaign, IL, USA; [b]Graduate School of Education, University of Pennsylvania, Philadelphia, PA, USA

ABSTRACT
Learning analytics research has used both knowledge engineering and machine learning methods to model student behaviors within the context of digital learning environments. In this paper, we compare these two approaches, as well as a hybrid approach combining the two types of methods. We illustrate the strengths of each approach in the context of a case study in building models able to detect when students "game the system", a behavior in which learners abuse the environment's support functionalities in order to succeed by guessing or copying answers. We compare the predictive performance, interpretability and generalizability of models created using each approach, doing so across multiple intelligent tutoring systems. In our case study, we show that the machine-learned model required less resources to develop, but was less interpretable and general. In contrast, the knowledge engineering approach resulted in the most interpretable and general model. Combining both approaches in a hybrid model allowed us to create a model that performed best across the three dimensions, but requiring increased resources to develop.

ARTICLE HISTORY
Received 24 October 2018
Accepted 18 April 2019

KEYWORDS
Learning analytics; student behavior modeling; predictive modeling; machine learning; knowledge engineering; gaming the system

Introduction

In recent years, increased usage of digital learning environments has led to more collection of fine-grained logs of students' learning process (Baker & Siemens, 2014). Those logs provide researchers with detailed data about students' behaviors and performance inside the learning environments that can be used by learning analytics researchers to study learning and related processes (e.g. Blikstein, 2011; Kostyuk, Almeda, & Baker, 2018; Yu & Jo, 2014). At this point, there has been more than a decade of research using interaction logs to study students' learning behavior such as help-seeking in intelligent tutoring systems (Aleven, McLaren, Roll, & Koedinger, 2004; Vaessen, Prins, & Jeuring, 2014), self-explanation strategies (Shih, Koedinger, & Scheines, 2008) and the usage of different meta-cognitive strategies (Kinnebrew, Segedy, & Biswas, 2014). These models have been used for a range of purposes, from driving real-time intervention targeted towards making student behavior more effective and improving learning outcomes (Baker et al., 2006; Roll, Aleven, McLaren, & Koedinger, 2011), to being able to retrospectively study how learning behaviors develop, manifest, and impact learning (Kostyuk et al., 2018; Muldner, Burleson, Van de Sande, & VanLehn, 2011; Shih et al., 2008). This research has leveraged two broad classes of methods to model learning behaviors: knowledge engineering (Aleven et al., 2004; Muldner et al., 2011; Paquette, de Carvalho, & Baker, 2014; Shih et al., 2008), sometimes also referred to as rational modeling, and machine learning/data mining (Azevedo et al., 2009; Hsiao et al., 2018; Kinnebrew et al., 2014; Vaessen et al., 2014;

CONTACT Luc Paquette lpaq@illinois.edu

Yin, 2017). Both types of method are common in learning analytics research, but their trade-offs remain under-studied.

Using knowledge engineering, researchers develop models that are designed to reproduce the knowledge we have about a specific learning behavior. This is often achieved by designing a set of rules that matches a general common-sense definition of the behavior (e.g. Muldner et al., 2011) or by explicitly eliciting knowledge from an expert about how they determine whether a student is exhibiting a specific behavior (e.g. Paquette, de Carvalho, & Baker, 2014). A primary advantage of knowledge engineering is that, unlike machine learning, it does not require a large amount of coded data providing examples of students' behaviors since the knowledge is acquired directly from experts. On the other hand, eliciting knowledge from experts can be a challenging task as experts are sometimes unaware of their own knowledge that they use implicitly when making judgements (Berry, 1987), and small differences in the details of a knowledge engineered model can have major implications for inferences made using that model (Kovanovic et al., 2016).

Machine learning approaches attempt to resolve the challenge of implicit expertise by leveraging data-driven algorithms to discover models from positive and negative examples of a student behavior. Using this approach, a large amount of data is automatically inspected in order to find relationships between the students' fine-grained actions and higher-level behaviors, avoiding the need to explicitly elicit knowledge about the behavior. However, several learning analytics researchers have questioned the generalizability of machine learning findings to new learning contexts or populations (Gardner & Brooks, 2018; Ocumpaugh, Baker, Gowda, Heffernan, & Heffernan, 2014).

Despite the significant quantities of research using each of these categories of method, there has not been direct comparison of the relative advantages of these two approaches. In this paper, we present a case study in which we compare both approaches in the context of modeling "gaming the system" behavior. We synthesize and build on research we previously conducted in which we built models of gaming the system using knowledge engineering (Paquette, de Carvalho, & Baker, 2014), machine learning (Baker & de Carvalho, 2008) and a hybrid approach (Paquette, de Carvalho, Baker, & Ocumpaugh, 2014), in order to directly compare their relative advantages. We specifically compare the approaches across three main dimensions: (1) accuracy on the original data used to create them, (2) model interpretability (3) model generalizability to new data and new contexts.

Gaming the system

"Gaming the system" is a behavior that manifests when students attempt to complete learning tasks, in a digital learning environment, by exploiting properties of the environment rather than by attempting to learn (Baker, Corbett, Roll, & Koedinger, 2008). Gaming the system has been of particular interest to learning analytics researchers due to its relationship with poorer learning outcomes (Beck & Rodrigo, 2014; Cocea, Hershkovitz, & Baker, 2009; Fancsali, 2013; Pardos, Baker, San Pedro, Gowda, & Gowda, 2014) and lower long-term academic attainment (San Pedro, Baker, Bowers, & Heffernan, 2013).

Gaming behaviors have been relatively thoroughly studied in the context of intelligent tutoring systems. In that context, gaming behavior can take the form of an inappropriate help-seeking strategy where students abuse the help functionalities of the system (Aleven et al., 2004; Johns & Woolf, 2006; Muldner et al., 2011; Murray & VanLehn, 2005) by requesting hints until the system provides them with the answer, or by quickly requesting hints without first attempting to solve the problem on their own (Aleven et al., 2004). Students may also game the system by systematically attempting to guess the answer (Johns & Woolf, 2006; Muldner et al., 2011; Walonoski & Heffernan, 2006), for example, by entering sequences of similar answers or by copying elements from the text of the problem as their answers. In some systems, students also game by making intentional errors to obtain the answer (Baker, Mitrovic, & Mathews, 2010; Murray & VanLehn, 2005).

Modeling approaches such as machine learning and knowledge engineering have been used to develop models able to detect gaming behaviors. With machine learning, algorithms have been used to automatically find relationships between indicators of the students' behavior, computed using the interaction logs, and human judgments of gaming behavior collected through field observations or hand-coding of log data (Baker et al., 2008; Pardos et al., 2014). By contrast, in knowledge engineering, experts' knowledge has been used to develop rules that are indicative of behaviors associated with gaming the system such as, for example, quickly entering incorrect answers or engaging in different types of help abuse (Johns & Woolf, 2006; Muldner et al., 2011).

Modeling gaming behaviors

To compare the predictive performance, interpretability, and generalizability of three different modeling approaches (machine learning, knowledge engineering and a third "hybrid" approach that attempts to combine the strength of both), we created models able to detect gaming behaviors using each approach. Each model was created using the same dataset obtained from 59 students using Cognitive Tutor Algebra (Koedinger & Corbett, 2006) during an entire school year as part of their regular mathematics curriculum (Pittsburgh Science of Learning Center DataShop "Algebra I 2005–2006 (Hampton only)" dataset; Koedinger et al., 2010).

The Cognitive Tutor Algebra environment offers mathematical problems broken down into the steps of the process used to solve the problem. As the student works through the problem, the tutor offers immediate feedback indicating whether each problem-solving step was correct or incorrect. In addition, the tutor provides feedback messages for common misconceptions. Finally, Cognitive Tutors also offer multi-step hints, available at any time, which provide increasingly specific hints about what should be done next.

The dataset was segmented in sequences of five actions, called clips, illustrating the student's behavior. A total of 10,397 clips were randomly selected and coded by a human expert, previously found to have good inter-rater reliability with another human expert, as involving gaming the system or not involving gaming the system (Baker & de Carvalho, 2008). Overall, 708 clips were coded as gaming and 9689 as not gaming. The complete dataset was separated into two subsets, used to develop each of the models included in our case study. First, a training set was created using 75% of the data, 7798 clips (531 gaming and 7267 non-gaming). Then, the remaining 25% of the data, 2619 clips (177 gaming and 2422 non-gaming), was kept as a held-out test set to validate how well the models generalize to new, unseen data from the same context.

Machine learning

The first approach we investigate is machine learning. Although there are several types of machine learning, the most common approach used by learning analytics research attempting to model student behavior is supervised learning, also referred to as behavior detection or classification. A second common machine learning approach, discovering behavior categories bottom-up through cluster analysis, factor analysis, or principal component analysis, is also common in learning analytics research (e.g. Hsiao et al., 2018; Vaessen et al., 2014; Yin, 2017), but is typically not targeted towards modeling a specific construct and is therefore harder to compare directly to other approaches.

To use supervised learning, the model's designer must first create a set of features, variables that summarizes the students' behavior over the relevant period, a clip in our model of gaming the system. The values of those features are computed for each clip. Finally, a supervised learning algorithm is applied to automatically find the relationships that are indicative of whether a data point contains the behavior (identified in expert labels).

The machine-learned model of gaming the system that we used for our comparison was first published in (Baker & de Carvalho, 2008). Within this approach, 26 features were calculated for each clip. This included features about (1) the actions contained in the clips, such as the type of the actions (correct, incorrect, help request, etc.) and whether they were the student's first attempt at completing a step of the problem, (2) the system's assessment of the student's knowledge, computed using

Bayesian Knowledge Tracing (Corbett & Anderson, 1995), (3) the time spent on each step, absolute and relative, and (4) interactions that directly preceded the current actions, such as the number of help request in the last 8 actions and the number of errors in the last 5 actions.

The J48 decision tree algorithm, also referred to as C4.5 (Quinlan, 1993), achieved the best performance of several alternatives and was used to detect gaming behavior from those 26 features. J48 decision trees make a set of if–then-else (either-or) decisions, in order, ultimately resulting in a prediction and a percent confidence in that prediction. Depending on which path through the tree is chosen, a different set of if–then-else decisions will be made. J48 constructs a decision tree by iteratively selecting the feature that best splits the training data according to whether students were gaming or not, and then re-combining splits that do not end up creating enough differentiation in predictions to be worth retaining in the tree. The quality and performance of the model was evaluated using two metrics: Cohen's Kappa (Cohen, 1960), the degree to which the model is better than chance at detecting gaming behaviors (chance = 0, perfect = 1), and AUC ROC (Hanley & McNeil, 1982), the probability that, given an example of gaming behavior and one of non-gaming behavior, the model will correctly identify the example containing gaming (chance = 0.5, perfect = 1).

The original model achieved a performance of Kappa = 0.40 and AUC = 0.69 (Baker et al., 2008). However, its performance was evaluated on the same data that was used to develop the model, possibly overstating the model's quality. In order to better evaluate performance, this model was re-built using the training set (75% of the data) and re-validated using the test set (25% of the data). With this more stringent validation, the model achieved Kappa = 0.218 and AUC = 0.691.

Knowledge engineering

The second approach we studied for detecting gaming the system was knowledge engineering, where a model is created by developing rules that capture the knowledge required to identify the behavior. We developed the model (Paquette, de Carvalho, & Baker, 2014) by studying the process of an expert as she labeled clips of student actions, displayed through "text replay" visualizations of log data (Baker, Corbett, & Wagner, 2006), as gaming or not gaming. A knowledge engineer who was not involved in the development of the machine-learned model conducted a cognitive task analysis (Clark, Feldon, van Merriënboer, Yates, & Early, 2008; Cooke, 1994), using data from our training set, during which knowledge was elicited from the expert through a combination of observations and interviews with active participation from the elicitor (Cooke, 1994; Meyer, 1992).

Through observation of the expert's coding and interviews with the expert, it was determined that the coding of text replays is separated into two main processes: interpreting the student's actions, and using these interpretations to identify patterns of actions that act as indicators of gaming. An analysis of the expert's process of interpreting the student's actions identified 19 different constituents of gaming patterns (Paquette, de Carvalho, & Baker, 2014).

In general, a *constituent*, or unit of behaviors that triggered a substantive response from the expert, tended to be defined by how long the student paused before or after making an action in the system; the expert judged whether these pauses demonstrated reflection and engagement. For example, a long pause before a student makes another attempt might act as a cue that the student is thinking about their next step, and a long pause after a help request might indicate that the student is reading the help message. Conversely, a short pause after a help request might indicate that the student is trying to find the answer in a bottom-out hint provided by the system.

In addition to looking at the length of pauses before and after actions, the expert also identifies cues in the answers entered by the student. In particular, the expert looks at whether the same answer is used multiple times in different problem steps (parts of the user interface), and whether similar answers are inputted in sequence. The expert also examines whether the student is changing what problem step they are working on without successfully completing the current step (e.g. Baker et al., 2010).

The constituents identified by the knowledge engineer shared some similar themes with the features developed for the machine-learned model. Both the machine-learned and knowledge engineered model considered the type of the student's actions (e.g. errors and help requests), the time taken between actions and whether students persisted on the same steps (same element of the user interface) or attempted multiple different steps. However, the features in the machine-learned model also relied on information computed from past behavior of the student on similar step, such as the estimated probability that the student knew how to complete a step (computed using past successes and failures) and the average time spent by the student on comparable steps in the past. In contrast, the knowledge engineered model only relied on the localized information contained in a clip.

Beyond identifying constituents, experts also pay attention to patterns of behaviors that indicate gaming (Paquette, de Carvalho, & Baker, 2014). Through interviews where the expert explained the coding process, 13 substantive action patterns representing gaming behavior were identified. For example, some student might guess by entering a sequence of similar answers or quickly reusing the same answer across different problem steps.

It was found that using these 13 patterns enabled the creation of a model of gaming behaviors (Paquette, de Carvalho, & Baker, 2014). In this model, a clip of actions is judged as gaming behavior if it contains any of the 13 patterns. This model was validated using the same data set as used in (Baker & de Carvalho, 2008) and achieved a Kappa of 0.430 when applied to the training set and Kappa = 0.330 when applied to the held-out test set. No AUC values were computed for the knowledge engineered model since the model provides binary predictions (gaming or not gaming) but not confidence values for its predictions.

Hybrid approach

Third, we consider models developed using a hybrid approach, combining elements of both machine learning and knowledge engineering. As the performance of machine-learned models is highly dependent on creating a good set of features, the goal of this approach is to identify how the knowledge acquired through knowledge engineering can be leveraged to develop a set of features that are more representative of the student behavior being modeled than the features that would otherwise have been created without prior knowledge engineering.

In the context of our models of gaming the system, we created a machine-learned model that builds on the constituents we previously identified through knowledge engineering (Paquette, de Carvalho, Baker, & Ocumpaugh, 2014). More specifically, we developed an algorithm able to generate and filter a large number of action patterns similar to those identified in the knowledge engineered model. In total, 60 *pattern features* were selected and combined with a set of 25 *count features,* which tallied the number of times each of the 19 constituents were present in a given clip and how often actions within the clip were associated with 6 different action types.

Next a gaming detection model was created using four machine learning algorithms (Naïve Bayes, J48, JRip and Step regression), applied to the set of 85 features previously described (60 *pattern features* and 25 *count features*) and trained using the same 75% training set that was previously used. Naïve Bayes achieved the best performance out of the four algorithms tried. Naïve Bayes takes the probability of gaming versus not gaming based on each of the features, and then integrates the probability across all features, assuming conditional independence between features. When building the models, feature selection was executed using forward selection over the set of 85 features (60 pattern features and 25 count features). Performance of the models was evaluated on the previously defined test set; prior work on this model (Paquette, de Carvalho, Baker, & Ocumpaugh, 2014) used 6-fold student-level cross-validation but we use the same test set for all models in this paper for fair comparison.

The forward selection process selected 22 features, 20 of which were pattern features and only 2 of which were count features (see Paquette, de Carvalho, Baker, & Ocumpaugh, 2014 for a full list of features). This initial model, that we call *Hybrid-PF* (for Hybrid-PatternFeatures, as this model was trained using pattern features in addition to the count features), achieved high Kappa (0.384), but lower than expected AUC (0.720). The low AUC may be due to the binary nature of the pattern features: a clip either contains a pattern feature or it does not. As a result, the detector has a very high confidence that a clip contains gaming behaviors when it matches one or more pattern features, but has a confidence very close to 0 for any clip that does not.

For this reason, we created a second model, called *Hybrid-C* (for *Hybrid-Counts*). This model was created using the same process as *Hybrid-PF*, but excluded the pattern features and included only the 25 count features. The best model, created using the Naïve Bayes algorithm, selected 6 features (see Paquette, de Carvalho, Baker, & Ocumpaugh, 2014 for a list of the selected features) and achieved a lower Kappa (Kappa = 0.332), but greatly improved AUC (AUC = 0.865) on the test set.

In order to combine the strengths of both the *Hybrid-PF* and the *Hybrid-C* detectors, we created a third detector, *Hybrid-E* (for Hybrid-Ensemble), that ensembles the two previous detectors. In this model, confidence that a clip contains gaming behavior is computed by averaging the confidences of the *Hybrid*-PF and the *Hybrid*-C confidences for each clip. *Hybrid-E* achieved a performance of Kappa = 0.376 and AUC = 0.876 on the test set.

Comparison of approaches

We compare the performance of each model across three dimensions: how accurate they were at predicting gaming behaviors in the test portion (25%) of the original dataset (Cognitive Tutor Algebra), how easy they are to interpret, and how well they generalize to datasets from different intelligent tutoring systems.

Predictive performance on the original data

Out of all our models of gaming the system, the hybrid approach (*Hybrid-E)* resulted in the highest predictive performance, with Kappa = 0.376 and AUC = 0.876 (see Table 1). The knowledge engineered model achieved higher performance (Kappa = 0.330) than the machine learning model (Kappa = 0.218).

However, direct comparison of the performance of the machine learning and the knowledge engineering approaches can be misleading. With both approaches, spending more time and effort on developing the model might increase performance. For machine learning, additional time can be taken to create new features that might better predict gaming behaviors, and later, more intensive, efforts to develop machine-learned models of gaming behaviors in other tutors achieved higher cross-validated performance. For example, models of gaming in ASSISTments and SQL Tutor achieved Kappa = 0.370 and AUC = 0.802 (Pardos et al., 2014), and Kappa = 0.360 and AUC = 0.770 (Baker et al., 2010) respectively. Similarly, spending additional time when eliciting knowledge from the expert might have resulted in the identification of additional action patterns, thus further improving the predictive performance of the knowledge engineered model.

In contrast, direct comparison to the performance of the hybrid approach is fairer. Time and effort required to create the hybrid model can be separated in two separate components:

Table 1. Summary of the predictive performance for each model for Cognitive Tutor Algebra. All models were validated using the same held-out test set (25% of the data).

Model	Kappa	AUC
Machine *l*earned	0.218	0.691
Knowledge engineered	0.330	N/A
Hybrid-PF	0.384	0.720
Hybrid-C	0.332	0.865
Hybrid-E	0.376	0.876

(1) time spent on knowledge elicitation and (2) time spent on developing a method to automatically combine the elicited knowledge. Additional time spent on knowledge elicitation could result in improved performance for the hybrid model, but it would also increase the performance of the knowledge engineered model. The main strength of the hybrid model, which allows it to achieve performance beyond either the knowledge engineering or machine learning approaches, resides in how it leverages machine learning to automatically find more complex patterns that leverage the elicited knowledge.

Interpretability

In addition to being used to automatically detect gaming during learning, models of gaming the system can also be used to study the nature of gaming behavior. For such studies, having an interpretable model of gaming behaviors would be useful for understanding why specific actions are predicted to be gaming by the model.

The knowledge engineered model represents the result of explicit elicitation of the expert's knowledge, making it relatively easy to interpret. Each constituent has a precise meaning and each action pattern presents an interpretable sequence of actions associated with gaming behavior. Overall, by its nature, the knowledge engineered model is designed to be interpretable. However, it doesn't allow a researcher to study gaming behavior beyond what the expert was able to elicit.

The hybrid models use similar features to those used by the knowledge engineered model. For this reason, it is also relatively easy to interpret; the pattern features that were selected by the algorithm can be interpreted the same way as those in the knowledge engineered model. In addition, the hybrid models provide probability distributions that can be used to further understand the relationship between the selected features and the likelihood of gaming behavior. However, it is important to keep in mind that those probability distributions only indicate how the values of the features are associated with higher or lower likelihoods of gaming behaviors, unlike pattern features which explicitly describe gaming behavior. One advantage that the hybrid approach has over knowledge engineering is that the automated discovery of action patterns might identify patterns that the expert might not have been able to explicitly describe. However, such patterns are not guaranteed to directly represent an expert's interpretation of student behavior.

Interpretability of a fully machine-learned model is highly dependent on the algorithm used to create the model. In our study, the machine-learned model was created using a J48 decision tree. Decision trees can generally be inspected to understand their decision-making process. Each node can be inspected to look at which feature is used to make a decision and how this node partitions the feature into two subgroups. It is relatively easy to inspect small trees in their entirety. However, as a tree becomes more and more complex, fully inspecting it can become intractable. For example, our J48 model of gaming the system has a maximum height of 17, with 448 different nodes (200 of which are leaves). Although each node, or even each branch of the tree, can individually be interpreted, interpreting the tree in its entirety is unrealistic due to its high complexity. As such, our machine-learned model is much less interpretable than either our knowledge engineered or hybrid models.

In general, the interpretability of a machine-learned model will depend highly on the algorithm used to create the model. Classic regression models (e.g. logistic regression, linear regression, and step regression) can be interpreted by inspecting the list of variables contained in the model and their associated coefficients, providing insights about how those variables are related to the studied constructs (although caution must be exercised in cases with high collinearity). Similarly, decision trees and decision rules can be inspected to identify the thresholds that are important for the partition of the feature space, but they can become difficult to interpret the more complex they become (i.e. when a tree has a large number of branches or there are large numbers of decision rules). Other models, such as recurrent neural networks, support vector machines, or K*, can be more difficult to interpret.

Generalization to new data

Although detectors of student behaviors are usually validated on new and unseen data using a held-out test set, this validation does not establish how well a model will generalize to new student populations or new learning environments (see Ocumpaugh et al., 2014 for an example of limited generalizability across student populations). In this section, we evaluate how well each of our gaming detectors generalize to data collected from two new intelligent tutoring systems: the scatterplot lesson of Cognitive Tutor Middle School (Baker, Corbett, & Koedinger, 2004) and ASSISTments (Razzaq et al., 2005). In Paquette, Baker, de Carvalho, and Ocumpaugh (2015), we studied how our knowledge engineered and hybrid models generalize to those new environments. We summarize those results here and extend them by adding new results related to the generalization of the machine-learned model.

The scatterplot lesson of Cognitive Tutor Middle School (Koedinger & Corbett, 2006) was built using the same platform as Cognitive Tutor Algebra. As such, it provides similar functionalities such as immediate feedback, on demand next-step hints, and common misconception messages. The main differences between algebra and scatterplot are the mathematical domain being taught and the population of students (high school vs. middle school).

We applied our detectors to three datasets collected in the context of a study of inter-rater agreement between experts when coding text replays (Baker, Corbett, & Wagner, 2006). Gaming labels were collected from two experts (named here Expert #1 and Expert #2). Expert #1 labeled 595 clips, 29 of which were labeled as gaming and 566 as not gaming. Expert #2 labeled 592 clips, 33 as gaming and 559 as not gaming. We also created a third dataset containing only clips for which the two experts agreed. This agreement dataset contained 571 clips, 19 of which were labeled as gaming and 552 as not gaming. We applied each of our five models to those three datasets; performance is shown in Table 2.

We also applied our models to data collected from students using the ASSISTments system. Like Cognitive Tutor Algebra, ASSISTments is an intelligent tutor for mathematics problems. ASSISTments offers similar functionality as Cognitive Tutor Algebra, including immediate feedback, on demand next-step hints and common misconception messages. However, the structure and presentation of problems in ASSISTments significantly differs from Cognitive Tutor Algebra. In ASSISTments, students are presented with an “original” problem that can be solved in one action by entering the correct answer. If a student solves the problem on their first attempt, they do not need to complete the individual steps. However, students who do not provide the correct answer may be required to answer additional scaffolding questions which break down the original problem into component skills.

Due to the different problem structure used by ASSISTments, instead of defining clips as sequences of five actions in a row, clips were defined as containing all the actions required to solve a problem, starting from the first action of an original (non-scaffolding) problem and ending with the last step before the beginning of the next original problem.

From the set of all available clips, two datasets were randomly selected. Sample #1 included 520 clips of 1–25 actions and sample #2 included 543 clips of 4–25 actions (see Paquette et al., 2015 for

Table 2. Comparison of model performance across Cognitive Tutor Algebra and the scatterplot lesson of Cognitive Tutor Middle School.

	Cognitive Tutor Algebra (original context)		Scatterplot Expert #1		Scatterplot Expert #2		Scatterplot Agreement	
Model	Kappa	AUC	Kappa	AUC	Kappa	AUC	Kappa	AUC
Machine learned	0.218	0.691	0.116	0.659	0.102	0.586	0.112	0.655
Knowledge Engineered	0.330	N/A	0.459	N/A	0.479	N/A	0.483	N/A
Hybrid-PF	0.384	0.720	0.440	0.819	0.438	0.795	0.451	0.877
Hybrid-C	0.332	0.865	0.345	0.894	0.360	0.889	0.331	0.949
Hybrid-E	0.376	0.876	0.430	0.917	0.427	0.905	0.438	0.973

more details on the creation of those sets), avoiding very brief clips where gaming was unlikely. Sample #1 ($N = 520$) contained 18 gaming clips and 502 non-gaming clips and sample #2 ($N = 543$) contained 46 gaming clips and 497 non-gaming clips. We also applied our models to a dataset that combined sample #1 and sample #2. This combined set ($N = 1063$) contained 64 gaming clips and 999 non-gaming clips. Performance for each model is presented in Table 3.

As illustrated in Tables 2 and 3, the machine learning approach resulted in the poorest transfer across data sets, with a modest decrease in performance in the scatterplot datasets and a substantial decrease in the ASSISTments datasets. The hybrid models considerably outperformed the machine-learned model for both the scatterplot and ASSISTments data.

The knowledge engineering approach achieved the highest performance when applied to new contexts, achieving comparable to slightly better performance than the hybrid models despite achieving lower performance in the original context; suggesting that the knowledge engineered model's performance is more stable across contexts.

Conclusions and lessons learned

The results from our study provide insights to researchers and developers regarding the relative advantages of different student behavior modeling approaches. However, since those results were obtained through a case study of only one student behavior (gaming the system), further research will be necessary to explore how our results generalize to other student behaviors. Additionally, our work has focused on identifying constituents of the students' behavior that were specifically related to gaming the system. Future work could investigate whether a broader qualitative approach to the interpretation of the data can be used to identify general constituents of the students' behavior. Such constituents could potentially be reused to train a range of different predictive models, thus allowing us to take advantage of the strengths of the knowledge engineering and hybrid approaches for new problems with minimal time and effort.

In this section, we summarize the overall strengths and weaknesses of each approach (machine learning, knowledge engineering and hybrid) as observed in the context of the detection of gaming the system behavior, and provide lessons learned regarding when each approach might be most appropriate depending on the model's intended usage.

The machine learning approach

In our study, the main advantage of the machine learning approach consisted in the lower quantity of resources and effort required to develop the model. Once the labeled data has been collected, the process of developing a machine-learned model does not require access to an expert with a deep understanding of gaming the system like the knowledge engineered and the hybrid approaches do. Similarly, it does not involve a complex knowledge elicitation process that requires a lot of resources, both in terms of human resources (expert, knowledge elicitor, etc.) and time. Using the machine learning approach usually only requires people who can process the interaction logs to

Table 3. Comparison of model performance across Cognitive Tutor Algebra and ASSISTments.

	Cognitive Tutor Algebra (original context)		ASSISTments Sample #1		ASSISTments Sample #2		ASSISTments Combined	
Model	Kappa	AUC	Kappa	AUC	Kappa	AUC	Kappa	AUC
Machine learned	0.218	0.691	0.000	0.572	0.063	0.737	0.050	0.709
Knowledge Engineered	0.330	N/A	0.228	N/A	0.240	N/A	0.256	N/A
Hybrid-PF	0.384	0.720	0.075	0.565	0.285	0.694	0.248	0.665
Hybrid-C	0.332	0.865	0.124	0.890	0.156	0.763	0.173	0.810
Hybrid-E	0.376	0.876	0.121	0.892	0.246	0.803	0.235	0.829

engineer and compute features, which is also required by alternative approaches. Additionally, machine-learned models can be built using relatively user-friendly off-the-shelf tools, which greatly reduce the expertise required to develop them.

Despite this lower requirement in expertise and resources, in our study, machine learning achieved comparable performance in the original context to the knowledge engineering approach. However, it also produced a model that was more difficult to interpret and did not generalize as well to new contexts. Our study thus suggests that, when building a model, it is useful to consider if those two characteristics are desired. If the sole purpose of the model is to detect the behavior in a specific context, the machine learning approach could be preferable due to its lower resource requirement. If some interpretability is desired, more interpretable types of machine-learned model could be selected, but knowledge engineering is likely to lead to more interpretable models.

The knowledge engineering approach

In the context of our case study, the main strengths of the knowledge engineering approach were its robustness when generalizing to new contexts and its interpretability. However, those advantages came at the cost of increased development efforts, requiring considerable time investment from both an expert in the modeled student behavior (gaming the system) and an experienced knowledge engineer. In our study, our knowledge engineering approach was comprehensive and time-intensive, involving multiple cycles of interviews and model construction. Simply intuiting a model in a single step is unlikely to be sufficient to achieve comparable performance to what was obtained here.

The highly interpretable nature of the model we developed using knowledge engineering increases its usefulness for studying student gaming behaviors. However, it is important to note that the knowledge engineering approach does not provide insights into behaviors that the expert might have failed to describe during the knowledge elicitation process, something that both machine learning (when the model is interpretable) and hybrid approaches can do.

Finally, in our case study, the knowledge engineering approach showed relative robustness when the model was applied to new contexts. This result suggests that the knowledge engineering approach could be prioritized if it is expected that the model will eventually be applied to heterogenous datasets, whether across different populations of students or different learning environments. However, additional research will be required to confirm whether this robust model transfer occurs for other student behaviors as well.

The hybrid approach

In our case study, the hybrid approach performed well across all three of the studied dimensions: it achieved high performance in the original context, it is interpretable and it transferred comparably well across contexts. However, this approach's main weakness is the greater resources required to develop the models. As the hybrid approach heavily relies on the information acquired during the knowledge elicitation process, but goes beyond that, creating a hybrid model requires more effort than creating a knowledge engineered model. In addition to this effort, there are not currently off-shelf methods for creating these hybrid models. Development of the hybrid models required us to develop a unique solution to identify the relevant information collected through knowledge engineering and to create a method combining this information with machine learning processes.

As such, the hybrid approach appears to be mostly appropriate when high performance is required across all three dimensions. Otherwise, the tradeoff between development effort and model quality may not justify the modest improvements achieved for each dimension individually.

Overall, as learning analytics research studying student behaviors in digital learning environments continues to grow, the community will need to better understand the relative advantages of different modeling approaches. Selecting the right type of model can facilitate progress for the specific research questions, development goals, and program of research that an individual research group or learning system developer is pursuing. Our case study in the detection of gaming the system offers insights into the potential merits and drawbacks of knowledge engineering, machine learning and hybrid approaches. Our findings suggest that these methods may have different strengths and weaknesses depending on the model developer's goals for the models' usage: how broadly they will need to apply the model they develop, and whether they seek to understand the model itself or simply be able to use it to generate inferences for further analyses or interventions.

Acknowledgements

The data that support the findings of this study are available from the corresponding author, Luc Paquette, upon reasonable request.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This research was supported by Fonds de Recherche du Quebec Nature et Technologies (FQRNT) post-doctoral fellowship and by National Science Foundation (NSF) [grant number SBE-0836012].

Notes on contributors

Luc Paquette is an Assistant Professor in Curriculum & Instruction at the University of Illinois at Urbana-Champaign. His research focuses on the usage of machine learning, data mining and knowledge engineering approaches to analyze and build predictive models of the behavior of students as they interact with digital learning environments such as MOOCs, intelligent tutoring systems and educational games. He is interested in studying how those behaviors are related to learning outcomes and how predictive models of those behaviors can be used to better support the students' learning experience.

Ryan S. Baker is Associate Professor at the University of Pennsylvania, and Director of the Penn Center for Learning Analytics. His lab conducts research on engagement and robust learning within online and blended learning, seeking to find actionable indicators that can be used today but which predict future student outcomes. Baker has developed models that can automatically detect student engagement in over a dozen online learning environments, and has led the development of an observational protocol and app for field observation of student engagement that has been used by over 150 researchers in 4 countries. Predictive analytics models he helped develop have been used to benefit hundreds of thousands of students, over a hundred thousand people have taken MOOCs he ran, and he has coordinated longitudinal studies that spanned over a decade. He was the founding president of the International Educational Data Mining Society, is currently serving as Editor of the journal Computer-Based Learning in Context, is Associate Editor of two journals, was the first technical director of the Pittsburgh Science of Learning Center DataShop, and currently serves as Co-Director of the MOOC Replication Framework (MORF). Baker has co-authored published papers with over 300 colleagues.

ORCID

Luc Paquette http://orcid.org/0000-0002-2738-3190

References

Aleven, V., McLaren, B. M., Roll, I., & Koedinger, K. R. (2004). Towards tutoring help seeking: Applying cognitive modeling to meta-cognitive skills. In *Proceedings of the 7th international conference on intelligent tutoring systems* (pp. 227–239). Maceió: Brazil.

Azevedo, R., Witherspoon, A. M., Graesser, A. C., McNamara, D. S., Chauncey, A., Siler, E., … Lintean, M. C. (2009). MetaTutor: Analyzing self-regulated learning in a tutoring system for biology. In *Proceedings of the 14th international conference on artificial intelligence in education* (pp. 635–637). Brighton: UK.

Baker, R. S., Corbett, A. T., & Koedinger, K. R. (2004). Learning to distinguish between representations of data: A cognitive tutor that uses contrasting cases. In *Proceedings of the international conference of the learning sciences* (pp. 58–65). Los Angeles: USA.

Baker, R. S., Corbett, A. T., Koedinger, K. R., Evenson, S., Roll, I., Wagner, A. Z., … Beck, J. E. (2006). Adapting to when students game an intelligent tutoring system. In *Proceedings of the 8th international on intelligent tutoring systems* (pp. 392–401). Jhongli: Taiwan.

Baker, R. S. J. D., Corbett, A. T., Roll, I., & Koedinger, K. R. (2008). Developing a generalizable detector of when students game the system. *User Modeling and User-Adapted Interaction*, *18*(3), 287–314.

Baker, R. S. J. D., Corbett, A. T., & Wagner, A. Z. (2006). Human classification of low-fidelity replays of student actions. In *Proceedings of the educational data mining workshop at intelligent tutoring systems 2006* (pp. 29–36). Jhongli: Taiwan.

Baker, R. S., & de Carvalho, A. M. J. A. (2008). Labeling student behavior faster and more precisely with text replays. In *Proceedings of the 1st international conference on educational data mining* (pp. 38–47). Montreal: Canada.

Baker, R. S. J. D., Mitrovic, A., & Mathews, M. (2010). Detecting gaming the system in constraint-based tutors. In *Proceedings of the 18th international conference on user modeling, adaptation and personalization* (pp. 267–278). Big Island Hawaii: USA.

Baker, R., & Siemens, G. (2014). Educational data mining and learning analytics. In R. Keith Sawyer (Ed.), *Cambridge handbook of the learning sciences* (2nd ed, pp. 253–274). Cambridge: Cambridge University Press.

Beck, J., & Rodrigo, M. M. T. (2014). Understanding wheel spinning in the context of affective factors. In *Proceedings of the 14th international conference on intelligent tutoring systems* (pp. 162–167). Honolulu: USA.

Berry, D. C. (1987). The problem of implicit knowledge. *Expert Systems*, *4*(3), 144–151.

Blikstein, P. (2011). Using learning analytics to assess students' behavior in open-ended programming tasks. In *Proceedings of the 1st international conference on learning analytics and knowledge* (pp. 110–116). Banff: Canada.

Clark, R. E., Feldon, D., van Merriënboer, J., Yates, K., & Early, S. (2008). Cognitive task analysis. In J. Michael Spector, M. David Merrill, Jan Elen, & MJ Bishop (Eds.), *Handbook of research on educational communications and technology* (3rd ed, pp. 575–593). New York: Springer-Verlag.

Cocea, M., Hershkovitz, A., & Baker, R. S. J. D. (2009). The impact of off-task and gaming behaviors on learning: Immediate or aggregate? In *Proceedings of the 14th international conference on artificial intelligence in education* (pp. 507–514). Brighton: UK.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, *20*(11), 37–46.

Cooke, N. J. (1994). Varieties of knowledge elicitation techniques. *International Journal of Human-Computer Studies*, *41*, 801–849.

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

Fancsali, S. E. (2013). Data-driven causal modeling of "gaming the system" and off-task behavior in cognitive tutor algebra. In *NIPS workshop on data driven education*. Lake Tahoe: USA.

Gardner, J., & Brooks, C. (2018). Student Success prediction in MOOCs. *User Modeling and User-Adapted Interaction*, *28*(2), 127–203.

Hanley, J., & McNeil, B. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, *143*, 29–36.

Hsiao, C. C., Huang, J. C., Huang, A. Y., Lu, O. H., Yin, C. J., & Yang, S. J. (2018). Exploring the effects of online learning behaviors on short-term and long-term learning outcomes in flipped classrooms. *Interactive Learning Environments*, 1–18.

Johns, J., & Woolf, B. P. (2006). A dynamic mixture model to detect student motivation and proficiency. In *Proceedings of the national conference on artificial intelligence* (pp. 163–168). Boston: USA.

Kinnebrew, J. S., Segedy, J. R., & Biswas, G. (2014). Analyzing the temporal evolution of students' behaviors in open-ended learning environments. *Metacognition and Learning*, *9*, 187–215.

Koedinger, K. R., Baker, R. S., Cunningham, K., Skogsholm, A., Leber, B., & Stamper, J. (2010). A data repository for the EDM community: The PSLC DataShop. In C. Romero, S. Ventura, M. Pechenizkiy, & R. S. J. d. Baker (Eds.), *Handbook of educational data mining* (pp. 43–55). Boca Raton: CRC Press.

Koedinger, K. R., & Corbett, A. (2006). Cognitive tutors: Technology bringing learning sciences to the classroom. In R. Keith Sawyer (Ed.), *The Cambridge handbook of the learning sciences* (pp. 61–77). Cambridge: Cambridge University Press.

Kostyuk, V., Almeda, M. V., & Baker, R. S. (2018). Correlating affect and behavior in reasoning mind with state test achievement. In *Proceedings of the international conference on learning analytics and knowledge* (pp. 26–30). Sydney: Australia.

Kovanovic, V., Gasevic, D., Dawson, S., Joksimovic, S., Baker, R. S., & Hatala, M. (2016). Does time-on-task estimation matter? Implications on validity of learning analytics findings. *Journal of Learning Analytics*, *2*(3), 81–110.

Meyer, M. A. (1992). How to apply the anthropological technique of participant observation to knowledge acquisition for expert systems. *IEEE Transactions on Systems, Man, and Cybernetics*, *22*, 983–991.

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.

Murray, R. C., & VanLehn, K. (2005). 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: Netherlands.

Ocumpaugh, J., Baker, R., Gowda, S., Heffernan, N., & Heffernan, C. (2014). Population validity for educational data mining models: A case study in affect detection. *British Journal of Educational Technology*, *45*(3), 487–501.

Paquette, L., Baker, R. S., de Carvalho, A. M. J. A., & Ocumpaugh, J. (2015). Cross-system transfer of machine learned and knowledge engineered models of gaming the system. In *Proceedings of the 23rd conference on user modeling, adaptation and personalization* (pp. 183–194). Dublin: Ireland.

Paquette, L., de Carvalho, A. M. J. A., & Baker, R. S. (2014). Towards understanding export coding of student disengagement in online learning. In *Proceedings of the 36th annual cognitive science conference* (pp. 1126–1131). Quebec: Canada.

Paquette, L., de Carvalho, A. M. J. A., Baker, R. S., & Ocumpaugh, J. (2014). Reengineering the feature distillation process: A case study in the detection of gaming the system. In *Proceedings of the 7th international conference on educational data mining* (pp. 284–287). London: UK.

Pardos, Z. A., Baker, R. S., San Pedro, M. O. C. Z., Gowda, S. M., & Gowda, S. M. (2014). Affective states and state tests: Investigating how affect and engagement during the school year predict end of year learning outcomes. *Journal of Learning Analytics*, *1*(1), 107–128.

Quinlan, R. (1993). *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann Publishers.

Razzaq, L., Feng, M., Nuzzo-Jones, G., Heffernan, N. T., Koedinger, K. R., Junker, B., … Livak, T. (2005). The Assistment project: Blending assessment and assisting. In *Proceedings of the 12th annual conference on artificial intelligence in education* (pp. 555–562). Amsterdam: Netherlands.

Roll, I., Aleven, V., McLaren, B. M., & Koedinger, K. R. (2011). Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system. *Learning and Instruction*, *21*(2), 267–280.

San Pedro, M. O. Z., Baker, R. S. J. D., Bowers, A. J., & Heffernan, N. T. (2013). Predicting college enrolment from student interaction with an intelligent tutoring system in middle school. In *Proceedings of the 6th international conference on educational data mining* (pp. 177–184). Memphis: USA.

Shih, B., Koedinger, K. R., & Scheines, R. (2008). A response time model for bottom-out hints as worked examples. In C. Romero, S. Ventura, M. Pechenizkiy, & R. S. J. d. Baker (Eds.), *Handbook of educational data mining* (pp. 201–212). Boca Raton: CRC Press.

Vaessen, B. E., Prins, F. J., & Jeuring, J. (2014). University students' achievement goals and help-seeking strategies in an intelligent tutoring system. *Computers & Education*, *72*, 196–208.

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

Yin, C. (2017). How to mine student behavior patterns in the traditional classroom. In *Proceedings of the 2017 international conference on advanced technologies enhancing education* (pp. 103–106). Qingdao: China.

Yu, T., & Jo, I. H. (2014). Educational technology approach toward learning analytics: Relationship between student online behavior and learning performance in higher education. In *Proceedings of the fourth international conference on learning analytics and knowledge* (pp. 269–270). Indianapolis: USA.