Iterative Feature Engineering Through Text Replays of Model Errors

Stefan Slater
University of Pennsylvania
slater.research@gmail.com

Ryan S. Baker
University of Pennsylvania
rybaker@upenn.edu

Yeyu Wang
University of Wisconsin – Madison
wangyeyu215@gmail.com

ABSTRACT
Feature engineering, the construction of contextual and relevant features from system log data, is a crucial component of developing robust and interpretable models in educational data mining contexts. The practice of feature engineering depends on domain experts and system developers working in tandem in order to creatively identify actions and behaviors of interest. In this paper we outline a method of iterative feature engineering using the misclassifications of earlier models. By selecting cases where earlier models and ground truth disagree, we can focus attention on specific behaviors, or patterns of behavior, that a model is not using in its predictions. We show that iterative feature engineering on cases of false positives and false negatives improved a model predicting quitting in an educational video game by 15%. We close by discussing applications of this method for addressing model performance gaps across different classes of learners, as well as precautions against model overfitting with using this method of feature engineering.

Keywords
Feature engineering, knowledge engineering, games, text replays

1. INTRODUCTION
Educational games and digital simulations are powerful educational tools that have seen increasing use in classrooms within the last decade. These digital environments afford students rich opportunities to engage deeply with content, adopt new and different identities [6], explore personally relevant domains [8], and develop non-cognitive skills such as productive persistence [17]. The adoption of educational games as tools for learning has been accompanied with an increasing focus on educational games as a medium for the application of educational data mining. The medium of educational games presents challenges for EDM methodologies, however, as the relative complexity of student behaviors in games can be quite broad when compared to more constrained environments such as intelligent tutoring systems (ITS).

Given the more complex behaviors possible for students in these environments, researchers studying learning in digital environments and games are able to identify and predict more complicated cognitive and non-cognitive constructs. Some examples of constructs identified in games include persistence [14], elegant problem solving [13], seriousness [5], carefulness [4], computational thinking [1], and mental demand [31].

This increased complexity places an increased importance on the feature engineering and/or knowledge engineering steps of the data science pipeline. Expert knowledge is often crucial for understanding specific patterns of behavior within educational games and simulations. For example, deep understanding of both gameplay design and conceptual understanding of physics were needed to develop a model of whether students had implicit conceptual understanding of physics based on how they responded to balls of different colors (connoting mass) in a physics game [22]. This understanding has driven feature engineering in many of these cases. Previous work by [23] has shown that feature selection and feature engineering of variables with high construct validity can lead to better model performance on unseen data. The question, then, is how we as researchers can quickly and effectively identify the specific patterns of player behavior that "matter" – how can we best separate the signal from noise in a large, complex dataset on student behavior and interaction?

Historically, social sciences researchers have addressed the complexity of human behaviors by combining qualitative methods providing "thick description" of actions [7] with quantitative methods to make scalable and general claims. However, the considerable amount of behavioral log data generated by modern learning systems poses a challenge to the qualitative analysis of human behaviors. One approach, termed "closing the interpretive loop" [24], is to refine and validate a model by looping back to the raw data, and checking whether the model and data are consistent. In an application of this method, [12] constructed a model to investigate how interactive indicators in the *Jaune Fluo* dataset relate to emotions in learning. By returning to and leveraging raw transcription data, they gained insights about micro-level interactions between speakers that could be used to drive modeling.

In this paper we propose a related approach -- a method for selecting specific cases of relevance from a larger dataset for further analysis, using instances of model mis-prediction. By adopting an iterative approach to model selection and feature engineering, we can use cases of false positives and false negatives to identify the specific cases where the model fails to accurately match student data, to better uncover relevant gameplay behaviors and patterns. We can then employ qualitative techniques to these cases to better understand what is occurring, and use these findings for additional feature engineering and model iteration. By closing the interpretive loop, we not only gain deeper understanding of the data, but also generate new contextual features for modeling in a way that is closely tied to observed patterns of behaviors in the data. We apply this method in the

Stefan Slater, Ryan Baker and Yeyu Wang "Iterative Feature Engineering Through Text Replays of Model Error" In: *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, Anna N. Rafferty, Jacob Whitehill, Violetta Cavalli-Sforza, and Cristobal Romero (eds.) 2020, pp. 503 - 508

broader context of studying student quitting behavior in the educational physics simulation game *Physics Playground* [28].

2. METHODS

Data for this work comes from a series of randomized controlled trials (RCTs) conducted at middle schools in Pennsylvania and Florida during the spring of 2019 using the educational physics simulation game *Physics Playground*, courtesy of the *Physics Playground* team**.**

Physics Playground teaches elementary physics concepts such as conservation of momentum and torque through a sandbox environment where players are tasked with drawing simple machines that move a ball to a balloon elsewhere in the level. Students receive badges for successfully solving levels, and are able to use these badges to unlock different types of music, custom balls, and other cosmetic changes within the game. *Physics Playground* also contains in-game hints and scaffolds, accessible through a help button on the UI.

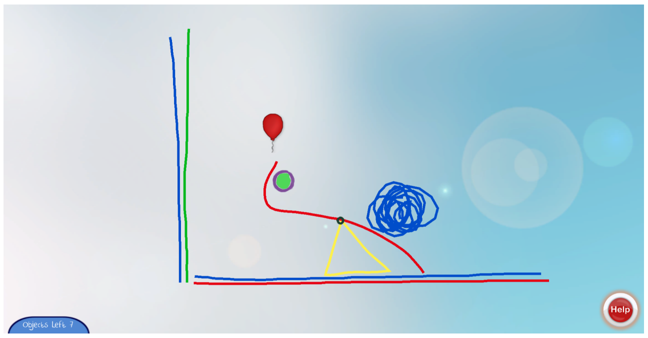

Figure 1. Physics Playground. The author has (unsuccessfully) built a lever and dropped a weight on it in an attempt to move the ball to the balloon. The help button (bottom right) and object counter (bottom left) are also pictured. The author would like to note that they are not a physicist.

A total of 96 students participated in the study. The RCTs were designed to test the effectiveness of several types of learning supports for Physics Playground on learning gains in the game. Students spent a total of ~110 minutes of class time playing Physics Playground in between a physics knowledge pretest and posttest, across four days. In the treatment condition, students were able to access a help button in the game UI that allowed the student to select multiple types of scaffolds to watch. Through the help button, students were able to receive help related to the use of game tools and mechanics, worked example solutions, and abstracted physics concepts. Students in the control condition were automatically prompted to use this button after three minutes had elapsed, but were unable to access the help button before that point. Preliminary analyses identified no significant differences in posttest scores or learning gains between conditions, so for the current study we combined these two groups and ignored condition assignment. Additional details on the study and its overall findings can be found in [29].

2.1 Data Structuring, Preprocessing, and Labeling

Gameplay data from the study were collected by the game's servers and output as .json files. A total of 703,765 records of student gameplay were collected during the study, where one record is a single logged student action in the game.

Several pre-processing steps were taken to prepare the data for analysis. Three students who did not complete the consenting process for the study were removed from the dataset. Events which occurred outside of study hours were also removed from the dataset. These events were due to students continuing to play the game in their free time. Attempts which were shorter than two seconds were also removed from the dataset. These attempts often consisted of students rapidly pressing the spacebar to reset their current level, without taking any in-game actions.

We also added additional contextual information into the dataset. We added the Physics Playground q-matrix into the dataset, which consists of the mapping between levels and physics constructs to be taught, as well as the simple machines associated with each level's solution. We added a series of session, visit, and attempt IDs to each record. A "session" is a length of time from student login to student logout. New sessions can begin when a student begins playing Physics Playground for the first time each day, or, when a student refreshes their browser. A total of 586 gameplay sessions were recorded, for an average of six sessions for each student. It is worth noting that students played the game for four days within the study; the higher average number of sessions is because students could accidentally refresh their browsers, or hit the "back" button, which began the logging of a new session. Within each session are "visits" – a visit lasts from the beginning of a level to the end of a level, whether the student solves that level successfully or quits to go to a different level. We identified 2906 total visits, with an average of 30 visits for each user – slightly less than the 34 levels available to play in the game for the current study. Finally, within each visit are "attempts" – an attempt begins any time that the level is initialized, and ends when a student either successfully solves the level, restarts the level, or quits the level. We identified 16,546 total attempts in the game, with an average of 172 for each user.

Given this structure of sessions, visits, and attempts, we defined a "quit event" as any time a student begins a new visit, within the same session, when their previous attempt was not successful. This represents a student failing to solve a level, leaving that level entirely, and playing a different level within the game. From each quit event, we labeled each record that happened up to 120 seconds before the event as "quit", and all other records as "not quit". Previous work on predicting quitting in Physics Playground used aggregations of 60-second clips within each attempt, e.g. [10]. In contrast, our method of labeling quitting at the event level, and up to 120 seconds prior, allows us to identify quitting across attempts, and sometimes across visits, in order to allow earlier detection and intervention by automated systems or in-classroom educators.

2.2 Initial Feature Engineering and Model Fitting

Drawing on previous literature that has explored Physics Playground [10,13], we developed an initial set of 32 features to use in predicting student quit behavior. These features included counts of each type of object or simple machine (weights, ramps, levers, pendulums, springboards, freeforms, and pins) that the student had drawn total and per attempt, the number of times students went some number of seconds without recording an action (5, 10, 15, 30, and 60 seconds), and whether students used each type of scaffold (worked examples, game tools, and physics animations) as well as the number of scaffolds that they used total and in each attempt. We also developed features to capture the amount of time that students spent using scaffolds, as well as the amount of time that had passed (in seconds) since the last time a scaffold was used. Finally, we recorded the elapsed time of each attempt, as well as the total elapsed time of the session, and the number of badges that students had earned so far.

For modeling quit behavior, we chose to use a relatively simple logistic regression model rather than more sophisticated algorithms such as a decision tree, gradient classifier, or recurrent neural network. Regression-based models are easier to implement into *Physics Playground's* Unity-based architecture than more sophisticated machine learning models. We used five-fold student-level cross validation in RapidMiner 9.4 [15]. We did not use any feature selection procedures for modeling; each feature was used as a component in the final model. We did not believe that our feature space was large enough to warrant feature selection. We used AUC ROC as our goodness metric, as we were more interested in overall model performance than optimizing our quit prediction threshold.

This initial model, which we will call the "original" model with "original" features in this paper, has an AUC of 0.688.

2.3 Error Identification and Feature Re-Engineering

Using the confusion matrix of this initial model, we identified all false positives and false negatives and mapped these events onto the attempts in the dataset. In other words, if *any* record within an attempt contained a case of model mis-prediction, we labeled the entire attempt as a mis-prediction. This resulted in 1,487 attempts labeled as cases of false negatives (9% of all attempts) and 298 cases of false positives (2% of all attempts). We then used text replays [2] to qualitatively code these attempts for patterns of engagement or behavior that we believed could be related to quitting behavior in players. Text replays have been used previously to conduct in-depth study of other constructs such as gaming the system [19], as well as to obtain training labels for the development of detectors [21, 23, 5]. [19]'s research shows that they can be a powerful tool for developing thick descriptions of learner behavior, and that this deeper understanding can lead to substantially better models of that behavior [18]. We randomly selected 100 examples each of false positives and false negatives for this coding process and conducted text replays on these attempts, taking notes on potential new features which could capture behaviors that we observed in the data. This coding procedure was done by a single researcher. As in [19], reliability measures were not obtained, as the goal was to develop new features that could be applied to the data programmatically rather than to develop a scalable human-based coding method. In our coding, we also viewed only single attempts, not looking at preceding or subsequent attempts (as in most prior uses of text replays).

Overfitting is an inherent concern for iterative feature engineering processes; we will discuss in the discussion section why overfitting may be particularly concerning for this paper's method. Because we wanted to overfit as little as possible, we only looked at text replays of false negatives and false positives. We intentionally did not view text replays of cases of true positives or true negatives. In other words, when we saw a behavioral pattern in false positives or false negatives, we did not double-check whether it was also seen in true positive or true negative cases, with a goal of deriving more features rather than attempting to conduct feature selection by hand by looking at the data (which could increase risk of over-fitting).

Our text replay and qualitative coding processes identified 14 additional features that we then developed software to apply to the dataset. Four of these new features related to scaffold use: **Multiple Uses Of Same Scaffold**, the number of times a student used the same learning support more than once in the same attempt; **Short Scaffold Time**, the number of times a student spent less than five seconds interacting with a scaffold; **Early Scaffold Use**, the number of times that a scaffold use appeared in the first third of actions that a student took in a given attempt; and **Multiple Scaffolds In Attempt**, the number of times that a student used more than one scaffold in the same attempt. Four features related to attempt duration: **Long Attempt Count**, the total number of attempts over three minutes; **Average Last Three Attempt Times**, the average duration of the last three attempts that a student had; **Attempt Time Standard Deviation**, the standard deviation of time across all student attempts so far; and **Previous Attempt Duration**, the duration of the attempt immediately before the current one. Three new features related to machine drawing and use: **Net Objects Drawn**, the number of objects a student drew on the current attempt minus the number of objects a student erased; **Time Spent Drawing**, the total elapsed time between the start and end of a student drawing a machine; and **Unexpected Machine Used**, whether a student drew a machine that was not associated with the knowledge component of the current level. We also created a feature for **Consecutive Nudges**, the number of consecutive times the student clicked on the ball to attempt to move it (cf. [9]), and a feature for **Recently Restarted**, whether the student restarted an attempt within the last 120 seconds. A restart is when a student unsuccessfully solves a level, but retries the same level rather than quitting and going to a new one.

The final re-design to our model, which we called **Quit Flush**, went beyond just creating a new feature. During coding, especially for false positives, we noticed that the model would continue to predict quitting after a quit event when the student did not subsequently quit. A student would begin a new attempt with the model already predicting that the student would quit. Then, some amount of time after the attempt had started, the quit prediction would drop off, and the student would go on to either restart the level or complete it successfully. We hypothesized that this was because the student may have quit in an earlier attempt, and the model had not yet caught up to the student's new behavioral patterns in a different visit. Therefore, we constructed a separate dataset, which we called the Quit Flush dataset. In this dataset, we reset the values of *all* features following a quit event, starting the model over again from a blank slate whenever a quit was identified.

Following this feature engineering process, we replicated the model fitting steps of the original model exactly. We also fit a series of models where we held out each new feature, to examine the performance gain from adding each feature into the new model.

3. RESULTS

3.1 Original Model vs. Enhanced Model

Our enhanced model, using all 14 newly engineered features (but not including the quit flush), produced an AUC of 0.812 – a gain of almost 0.10, and a 15% improvement over the original model. The enhanced model's performance is comparable to the best performing models developed by [10], even with the limitation of a relatively simple logistic regression rather than the more sophisticated classification algorithm used in that paper. We will call this model the “enhanced model” with “enhanced features” in this paper.

3.2 Enhanced Model vs. Quit Flush Model

The quit flush model, using all 14 newly engineered features and resetting all features’ values after a quit, produced an AUC of 0.616 – slightly worse than even the original model. The poor performance of the quit flush model suggests that student quitting events are “sticky”, and that moving between levels does not necessarily indicate that a student starts working productively.

An ROC curve comparison between the AUC of the original model, enhanced model, and quit flush model is given in Figure 2.

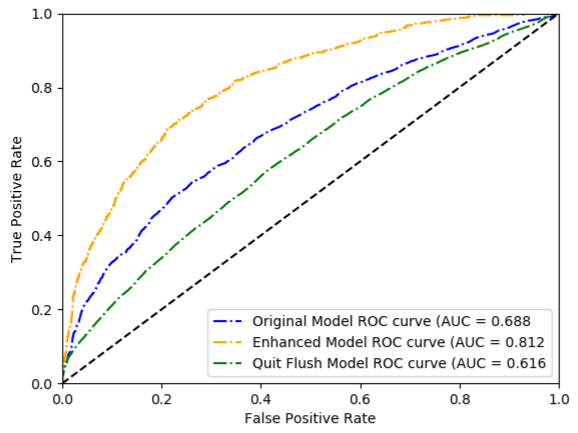

Figure 2. Comparison between Original Model AUC, Enhanced Model AUC, and Quit Flush Model AUC.

3.3 Leave One Out Model Results

Following our fitting of the enhanced model and the quit flush model, we then fit subsequent iterations of the enhanced model, holding one feature out each time. The purpose of this analysis was to identify the contribution that each individual feature made to the overall performance of the model.

Table 1. Comparison of feature impact on enhanced model performance.

Feature Name	AUC When Held Out	Delta
Consecutive Nudges	0.742	-0.070
Early Scaffold Used	0.793	-0.019
Previous Attempt Duration	0.812	--
Short Scaffold Use	0.812	--
Attempt Longer Than 3m	0.812	--
Average Last 3 Attempts	0.812	--
Uses of Same Scaffold	0.812	--
Multiple Scaffold Uses	0.812	--
Recent Restart	0.812	--
StD Attempt Duration	0.812	--
Time Drawing Objects	0.812	--
Unusual Object Used	0.812	--
Net Objects Drawn	0.812	--

We found that the performance increase of the enhanced model was driven primarily by just two features – **consecutive nudges**, and **early learning support use**. Student use of consecutive nudges was also found to be associated with student quitting in research by [9] – it is possible that students using nudges repeatedly could indicate that the student is trying to make an ineffective solution work when they cannot figure out a more productive means of solving the level. Further analysis of this behavior, perhaps incorporating qualitative interview data from students during or immediately after gameplay, could contribute to a better understanding of this behavioral pattern. Early use of learning support could either indicate a student who is completely stuck and doesn’t know where to start, or a lack of willingness to put in effort to solve a level, either of which could lead a student to quit.

4. DISCUSSION AND CONCLUSION

In this paper we demonstrate that iterative feature engineering using cases of model mis-prediction, conducting qualitative coding of text replays on instances of false positives and false negatives, can enhance model performance. We found that an “enhanced” model, using features that we developed through this method, performed 15% better than an “original” model on the same dataset. This performance gain came from just two of the 14 features that we iteratively engineered. We have several potential applications for this methodology, and an important caveat to make.

4.1 Applications of Iterative Feature Engineering

Educational data mining has been applied to a wide variety of problems, and we believe that iterative feature engineering may have specific learning contexts where it is more useful. Specifically, this technique relies on having rich, nuanced log data from which specific details of student interaction with the system can be drawn. For relatively simple contexts, such as students working on an online quiz system with very few choices or different components, this method may be more difficult to apply successfully, as there may not be enough variation in behavior for iterative feature engineering to be useful. On the other hand, contexts with rich contextual data may be better suited for this method. This method may be particularly useful for improving prediction models that were already developed using text replays [20, 11, 5, 23], though there is not a principled reason why the method could not be useful even for models initially developed using other methods. In applying iterative feature engineering to models not developed using text replays, it may be relevant to consider whether the behavior can be identified and understood from text replays – some forms of affect, for instance, may be difficult for humans to identify directly from this type of data.

4.2 Using Iterative Feature Engineering to Address Uneven Quality Across Populations

In this work, we applied our iterative feature engineering process to the entire dataset. However, recent papers have found that many EDM models can perform unevenly for different populations of learners, such as rural students versus urban students or non-native speakers versus native speakers (see review in [3]). We believe that the approach proposed in this paper may be a useful tool for fixing this type of inequity. Training separate models on each sub-population within the full dataset, and conducting qualitative coding on instances of error within these sub-population, could highlight different behavioral patterns seen for individuals in different groups. Population-specific features could then be engineered to equalize the performance of the model across groups of learners.

4.3 Caveat – The Potential for Overfitting

Consider this technique taken to its logical conclusion – monkeys on infinite typewriters, endlessly thresholding and re-thresholding features, defining and re-defining cutoffs, until the billionth permutation of this process produces a model with no error. This model, obviously, would be massively overfit and of next to no use in any broader context, such as on a new class of students. In practice, there are likely not enough educational data monkeys in the world to produce a model with no error. That said, conducting several cycles of iteration on the same dataset does run the risk of overfitting one's feature engineering process, and subsequently the model, to the particulars of the dataset being used. Therefore, this is likely a method best used only limited times over the course of model development. Ultimately, fully-withheld test datasets – or better yet, the collection of new datasets after the fitting process is complete -- should be used in final evaluation of a model (as seen in the trajectory of gaming the system modeling between [19] and [18]). It is not yet known how much iteration of this nature is beneficial, before diminishing returns or overfitting occur. It may be a valuable step for future research to investigate iterating multiple times and observing changes in model performance, identifying the elbow point for improvement. From the perspective of quantitative ethnography, researchers might consider stopping the iterative process when reaching *theoretical saturation*, seeing more data but failing to generate new insights [25].

4.4 Alternative Approaches to Feature Engineering and Text Replays

In this paper, we started with a model with initial features and then refined the model by examining the misclassified cases and deriving new features based on the qualitative interpretations of game play behaviors. However, this is not the only way that an iterative process of feature engineering could be conducted. [26] outline an alternative approach for constructing a theoretical-based and analytics-driven model by grounding analyses in qualitative data and exploring the pattern of data before model construction. Epistemic Network Analysis (ENA) models the co-occurrence of behaviors based on coded qualitative data and unpacks the complexity in the learning process [27]. Used first in epistemic games, other scholars have begun applying ENA techniques to this same problem. For example, [9] adopted ENA to explore why learners quit levels in Physics Playground, investigating cognitive processes based on student interaction with the game. Their study identified that students who crystalize their problem-solving strategy at the beginning of gameplay are more likely to quit the levels. This behavior pattern suggests new features to be engineered for the future study of quit model prediction.

4.5 Further Model Development and Future Directions

Poor performance of the quit flush model suggests that a student-level model may be beneficial to predicting quitting. Originally, our justification for creating the quit flush model was that we observed cases of quitting being predicted at the beginning of an attempt, and we hypothesized that this could be due to the model continuing to predict quitting behavior immediately after a quit event occurs. We anticipated that a quit flush feature would improve overall model performance by addressing these cases; however, the quit flush model performed significantly worse than both the enhanced and original models. Given this difference in performance, it is possible that an enhanced model which uses features aggregated across visits or even sessions of play, could be a more effective predictor than the simple count and duration features that we used in the current work. Previous work on *Physics Playground* has identified the existence of player typologies [30], representing distinct approaches that groups of players employ when playing *Physics Playground.* This work found that students who played *Physics Playground* could be assigned to one of three classes: *achievers,* motivated by in-game rewards such as badges; *explorers,* motivated by the ability to explore the game space, design interesting and unique machines, and push the boundaries of the physics simulation, and *disengaged* players, those players who did not engage with the game to the same degree as their peers. These classes of players could serve as valuable student-level features that inform overall patterns of play. Further work on this topic may also use features aggregated up to each level, as previous work has [10, 13]. While we did not use these features for the current work, because of difficulties in generating these variables at run-time inside the game environment, future work that is focused on using the quit model for analysis entirely outside of the game might benefit from using these various levels of aggregation.

This work has leveraged qualitative analysis in a somewhat different fashion than prior efforts within the EDM community. Qualitative techniques are not new within the fields of educational data mining and learning analytics. Text replays – human review of student behavior to generate labels – have been used for over a decade [2]. Similar work, such as [16] has focused on generating human-readable samples of student-tutor interactions. However, these methods are usually used to generate ground truth labels in order to construct models, or to better understand the relationships that these behaviors have with one another. In this paper, we utilize qualitative coding of student behaviors to develop features that subsequent models can be trained on. By bringing more qualitative understanding and analysis into educational data mining and learning analytics, and synthesizing these approaches with quantitative modeling practices, we can develop models that perform better and are more understandable by the research community.

5. ACKNOWLEDGMENTS

We would like to thank the Physics Playground team for their assistance in producing the datasets used.

6. REFERENCES

[1] Almeda, M.V., Rowe, E., Asbell-Clarke, J., Scruggs, R., Baker, R., Bardar, E., Gasca, S. (2019) Modeling Implicit Computational Thinking in Zoombinis Mudball Wall Puzzle Gameplay. To appear in *Proceedings of the 2019 Technology, Mind, and Society Conference*.

[2] Baker, R.S., & de Carvalho, A.M.J.A. (2008). Labeling Student Behavior Faster and More Precisely with Text Replays. *Proc. of the 1st Int'l Conf. on Educational Data Mining,* 38-47.

[3] Baker, R.S., Ogan, A.E., Madaio, M., Walker, E. (2019) Culture in Computer-Based Learning Systems: Challenges and Opportunities. *Computer-Based Learning in Context, 1*(1), 1-13.

[4] Banawan, M.P., Rodrigo, M.M.T., & Andres, J.M.A.L. (2015). An Investigation of Frustration Among Students Using Physics Playground. *Proceedings of the 23rd International Conference on Computers in Education.*

[5] DiCerbo, K., & Kidwai, K. (2013). Detecting player goals from game log files. *Proceedings of the 6th Annoal Conference on Educational Data Mining.*

[6] Gaydos, M.J., & Devane, B.M. (2019). Designing for identity in game-based learning. *Mind, Culture, and Activity, 26*(1), 61-74.

[7] Geertz, C. (1973). Thick description: Toward an interpretive theory of culture. In *The interpretation of cultures: Selected essays* (pp. 3–30). New York: Basic Books.

[8] Holbert, N., & Wilensky, U. (2019). Designing educational video games to be objects-to-think-with. *Journal of the Learning Sciences, 28*(1), 32-72.

[9] Karumbaiah, S., Baker, R.S., Barany, A., Shute, V. (2019) Using Epistemic Networks with Automated Codes to Understand Why Players Quit Levels in a Learning Game. *Proc. of the 1st Int'l Conference on Quantitative Ethnography*, 106-116.

[10] Karumbaiah, S., Baker, R.S., & Shute, V. (2018). Predicting Quitting in Students Playing a Learning Game. *Proceedings of the 11th International Conference on Educational Data Mining,* 21-31.

[11] Lee, D.M.C., Rodrigo, M.M.T., Baker, R.S., Sugay, J.O., & Coronel, A. (2011). Exploring the relationship between novice programmer confusion and achievement. *Proceedings of the 2011 International Conference on Affective Computing and Intelligent Interaction*, 175-184.

[12] Lund, K., Quignard, M., & Shaffer, D.W. (2017). Gaining Insight by Transforming between Temporal Representations of Human Interaction. *Journal of Learning Analytics, 4*(3), 102-122.

[13] Malkiewich L.J., Baker, R.S., Shute, V., Kai, S., Paquette, L. (2016). Classifying behavior to elucidate elegant problem solving in an educational game. *Proceedings of the 9th International Conference on Educational Data Mining,* 448-453.

[14] Malkiewich, L.J., Lee, A., Slater, S., Xing, C., & Chase, C.C. (2016). No Lives Left: How Common Game Features Could Undermine Persistence, Challenge-Seeking and Learning to Program. *Proceedings of the 2016 International Conference of the Learning Sciences,* 186-193.

[15] Mierswa, I., & Klinkenberg, R. (2019). RapidMiner Studio (9.4). Retrieved from https://rapidminer.com.

[16] Mostow, J., Beck, J., Cen, H., Cuneo, A., Gouvea, E., & Heiner, C. (2005). An educational data mining tool to browse tutor-student interactions: Time will tell. *Proceedings of the Workshop on Educational Data Mining, National Conference on Artificial Intelligence ,* 15-22.

[17] Owen, V.E., Roy, M.H., Thai, K.P., Burnett, V., Jacobs, D., Keylor, E., & Baker, R.S. (2019). Detecting Wheel-Spinning and Productive Persistence in Educational Games. *Proc. of the 12th International Conf. on Educational Data Mining,* 378-383.

[18] Paquette, L., Baker, R.S. (in press) Comparing machine learning to knowledge engineering for student behavior modelling: A case study in gaming the system. To appear in *Interactive Learning Environments*.

[19] 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. *Proc. of the 7th Int'l Conf on Educational Data Mining,* 284-287.

[20] Richey, J.E., Andres-Bray, J.M.L., Mogessie, M., Scruggs, R., Andres, J.M.A.L., Star, R.J., Baker, R.S., McLaren, B.M. (in press). More Confusion and Frustration, Better Learning: The Impact of Erroneous Examples. To appear in *Computers and Education.*

[21] Rodrigo, M.M.T., Baker, R.S.J.d., McLaren, B., Jayme, A., Dy, T. (2012) Development of a Workbench to Address the Educational Data Mining Bottleneck. *Proceedings of the 5th International Conference on Educational Data Mining*, 152-155.

[22] Rowe, E., Asbell-Clarke, J., Baker, R.S., Eagle, M., Hicks, A.G., Barnes, T.M., Brown, R.A., Edwards, T. (2017) Assessing Implicit Science Learning in Digital Games. *Computers in Human Behavior*, 76C, 617-630.

[23] Sao Pedro, M.A., Baker, R.S.J.d., Gobert, J., Montalvo, O. Nakama, A. (2013) Leveraging Machine-Learned Detectors of Systematic Inquiry Behavior to Estimate and Predict Transfer of Inquiry Skill. *User Modeling and User-Adapted Interaction, 23* (1), 1-39.

[24] Shaffer, D. W. (2017). Quantitative ethnography. Madison, WI: Cathcart Press.

[25] Shaffer, D. W. (2018). Big data for thick description of deep learning. In K. Millis, D. Long, J. Magliano, and K. Weimer (Eds.), Deep learning: Multi-disciplinary approaches (pp. 262-275). NY, NY: Routledge.

[26] Shaffer, D. W. & Ruis, A. R. (2017). Epistemic network analysis: A worked example of theory-based learning analytics. In C. Lang, G. Siemens, A. Wise, & D. Grasevic (Eds.), *Handbook of Learning Analytics* (pp. 175–187). Society for Learning Analytics Research.

[27] Shaffer, D. W., Collier, W., & Ruis, A. R. (2016). A tutorial on epistemic network analysis: Analyzing the structure of connections in cognitive, social, and interaction data. *Journal of Learning Analytics, 3*(3), 9–45.

[28] Shute, V. J., Almond, R. G., & Rahimi, S. (2019). Physics Playground (Version 1.3) [Computer software]. Tallahassee, FL: Retrieved from https://pluto.coe.fsu.edu/ppteam/pp-links/

[29] Shute, V., Rahimi, S., & Smith, G. (2019). Game-Based Learning Analytics in Physics Playground. In *Data Analytics Approaches in Educational Games and Gamification Systems* (pp. 69-93). Springer, Singapore.

[30] Slater, S., Bowers, A., Kai, S., & Shute, V. (2017). A Typology of Players in the Game Physics Playground. *Proceedings of the 2017 DiGRA International Conference*.

[31] [32] Wiggins, J.B., Kulkarni, M., Min, W., Mott, B., Boyer, K.E., Wiebe, E., & Lester, J. (2018). Affect-based Early Prediction of Player Mental Demand and Engagement for Educational Games. *Proceedings of the 14th Artificial Intelligence and Interactive Digital Entertainment Conference*, 243-249.