Integrating Large Language Models and Machine Learning to Detect Struggle in Educational Games

Xiner Liu[1[0009-0004-3796-2251]], Zhanlan Wei[1[0009-0002-3931-6398]], Ryan S. Baker[1[0000-0002-3051-3232]], Shari J. Metcalf[3[0000-0002-7374-3249]], Jiayi Zhang[1[0000-0002-7334-4256]], Amanda Barany[1[0000-0003-2239-2271]], Stefan Slater[1[0000-0002-1016-1516]], Luke Swanson[2[0000-0002-8455-7441]], and David J. Gagnon[2[0000-0001-5126-0476]]

[1] University of Pennsylvania, Philadelphia PA 19102, USA
[2] University of Wisconsin, Madison WI 53711, USA
[3] Harvard University, Cambridge MA 02138, USA
xiner@upenn.edu

Abstract. This study explores the integration of Large Language Models (LLMs), specifically GPT-4o, with machine learning (ML) to automatically detect struggle behaviors in a science exploration game. We evaluate GPT-4o's ability to analyze text replays, which convert raw gameplay logs into human-readable sequences of player actions, against existing machine-learned struggle detectors. The LLM and ML approaches achieve comparable performance but have complementary strengths. GPT-4o is effective at identifying struggle in instances where the human-engineered features used in traditional ML models fail to capture meaningful behavioral patterns, while ML detectors perform better at identifying struggle in tasks that involve numerical reasoning (e.g., frequent pauses) and in more complex tasks where structured features provide clearer signals.

Keywords: Behavior prediction, Large language models, GPT, Text replay

1 Introduction and Related Work

Educational games have been shown to support learning by immersing students in interactive environments (e.g., [7]). They are well-suited for teaching skills that benefit from practical application (e.g., solving mathematical problems [18], understanding scientific phenomena [1], or exploring historical events [14]). Scholars have long recognized the importance of understanding students' emotions and behaviors during gameplay, which influence how students interact with the game [22] and shape their learning outcomes [19, 23]. Automated detectors have been developed to identify emotions and behaviors from real-time gameplay (e.g., [6, 25]), sometimes using physical sensors (e.g., [10]). However, these detectors have been expensive and time-consuming to develop and rarely scale across systems (though outside of game context, see [17]).

Recently, the emergence of large language models (LLMs) like GPT have opened new possibilities for analyzing textual data at scale in educational contexts (e.g., [2]).

Prior research suggested that LLMs can reliably process text for various tasks such as identifying thematic patterns [5], automating assignments grading [15], and providing feedback on written responses [24]. However, LLMs cannot currently directly ingest the types of interaction data that detectors have worked from. To use this new generation of methods, we must find a way to translate interaction data into a form that LLMs can easily understand and corresponds to something in their training corpuses.

One promising approach draws on a method previously used by researchers to label interaction data for training detectors. Humans, like LLMs, cannot easily read interaction data. Therefore, Baker et al. [4] proposed re-representing interaction data in an easy-to-read format, text replays—readable, sequential narratives of students' actions derived from interaction logs. Text replay has been used by teams of coders to generate ground truth labels for training ML models to detect specific behaviors (e.g., [3, 8, 11, 16, 21]). They allow rapidly identification of the behavior of interest (2-6 times faster than classroom or video observation) while achieving inter-rater agreement only slightly lower than classroom observation [4]. The formatted narratives help researchers recognize patterns that would be difficult to detect from raw data alone.

Text replay, at its core, is a textual representation of interaction data, and it is possible that LLMs may be able to read text replays just as humans can. In this study, we examine whether GPT-4o can detect moments of struggle in the science exploration game *Wake* using text replays. Our goal is to assess the feasibility of this approach and to compare its performance with more traditionally machine-learned detectors.

2 Methodologies and Results

Wake: Tales from the Aqualab is a science education game designed for middle school students in grades 6–9. In the game, players take on the role of a young marine biologist named Olivia to learn more about marine biomes and their respective ecological systems in the game. Each ecosystem includes multiple research sites where players take on "jobs." A "job" represents a scientific assignment or mission within the game, such as studying the impact of an invasive species or monitoring biodiversity in a specific habitat. *Wake* uses the Open Game Data infrastructure to track and log player activities [9]. The opengamedata-unity package connects the game to a cloud-based server, which records player actions throughout each session. The game organizes telemetry data into three categories: *Player Actions* (general navigations, e.g., entering data into scientific tools). *System Events* (system messages, e.g., narrative elements that direct the player's actions). *Progression Events* (in-game milestones, e.g., player completing tasks). The telemetry data records 33 distinct player actions, 12 system events, and 6 progression events. Each event includes metadata that specifies the timing, sequence, player identification, and game state at the time of the event. These events are transmitted to the Open Game Data server in real time creating a chronological log of gameplay.

The data used in the study were sampled from data collected from January 2024 to May 2024. This dataset includes 19,186 players across 42,889 sessions. We sampled data in three rounds. In the first round, we selected 200 cases (40 per month) to evaluate how well GPT and ML could detect struggle for *Wake*. Based on what we learned, we refined the instructions given to GPT. The second round included 100 new cases (20

per month) to check if the updated prompt caused overfitting and if GPT and ML could be integrated for more accurate struggle detection. The third round added another 100 cases to test whether the hybrid GPT-ML model could generalize to unseen data. We only included jobs that lasted at least five minutes and had over 100 logged actions to make sure there was enough gameplay data for analysis.

2.1 Creating and Coding Text Replay Clips

We then converted log data into text replays to facilitate the examination and coding of student struggle behaviors. Each replay included all actions from when a player accepted a job until they moved to a different job. Log events unrelated to gameplay or job completion were removed. Using timestamps from the logging system, we calculated the time elapsed since the start of each gameplay session (defined as a continuous period between entering and exiting the game). Elapsed time was rounded to the nearest second and listed cumulatively. If multiple events had the same timestamp, they were grouped under the same time to reflect co-occurrence. Session numbers were recorded to separate distinct play periods, and elapsed time was reset at the start of each session. Each text replay began with the job name, which allowed coders to cross-reference it with the *teacher guide* (sites.google.com/wisc.edu/waketeacherguide) that details each job's objectives and expected progression as context for identifying student struggling.

Evaluating the performance of GPT requires reliable ground truth labels. After generating the text replays, two graduate students with expertise in qualitative coding independently labeled each clip as "Struggle" or "Not struggle," based on all actions within each clip. Struggle was defined as any indicator where the player might be facing difficulties or in need of help. The definition was intentionally simplistic and broad, as the goal was not to exhaustively list all possible manifestations of struggle. Instead, both coders were encouraged to evaluate each case individually while considering the specific context in which the struggle emerged. Differences in the coders' game experience led to low initial inter-rater reliability ($\kappa = 0.45$) after round 1. Discrepancies between coders were resolved through moderated discussions to reach a consensus. After discussions, their agreement improved to $\kappa = 0.76$ in round 2 and 0.78 in round 3.

2.2 Re-construction the Machine Learned Struggle Detection

As part of efforts to improve student engagement and provide timely in-game scaffolding, a previous study developed a machine-learned detector to automatically identify instances of struggle within the game [12]. This serves as an ideal benchmark for evaluating the effectiveness of GPT in detecting struggle. To accommodate updates in the game's data logging structure, we re-created 58 features from that previous study. The features were aggregated, selected, and validated following the same approach in [12].

2.3 Automated Coding of Struggle with GPT

We coded the 200 text replay clips extracted from round 1 using GPT via OpenAI's application programming interface (API). For this study, we used the GPT-4o-2024-

11-20 model with all hyperparameters set to default values, except for the temperature, which was set to 0 to increase output consistency. Each prompting approach was replicated three times on a 10% subset (20 clips) of the round 1 data, with no more than two cases showing disagreement across the three runs. We then applied each prompt to code all 200 clips and calculated agreement between GPT's predicted struggle labels and the human-coded ground truth (see Table 1).

1. **Partially Knowledge Engineered Prompt.** We first assess GPT's ability to analyze student interactions and identify struggle based on the same broad definition outlined above. This setup incorporates a partially knowledge-engineered framework, rather than relying solely on the model's reasoning abilities, to reduce the likelihood that GPT would produce inconsistent or irrelevant interpretations of gameplay behavior.
2. **Incorporating Contextual Information.** Human coders rely on implicit knowledge to identify struggles in text replays. GPT, however, does not have access to this contextual knowledge by default. Without such information, GPT might misinterpret behaviors that would otherwise clearly indicate a struggle. Therefore, contextual details (specifically, job objectives, task difficulty ratings, and the expected progression of actions) were added to the text replay clips as a second prompting method.
3. **Annotated Examples with Human Explanations.** Earlier strategies relied solely on GPT's interpretation, this approach, however, introduced four annotated examples with human-provided explanations. These examples were drawn from a separate dataset and provided explicit instances of struggle with rationales. The examples illustrated situations where students 1) paused extensively during an experiment 2) frequently returned to previously visited locations, 3) repeatedly performed actions that deviated from the hints and 4) cycled through the same choices in experiments.

When annotated examples were included in the prompt, GPT performed comparably to the XGBoost model in detecting struggle. This prompting approach was then selected for the next stage of research. However, incorporating contextual information led to a decline in the model's performance, which aligns with previous research suggesting that excessive information can overwhelm GPT during decision-making [13].

Each job in the game is rated by the designers for difficulty across three core dimensions, *Experimentation*, *Argumentation*, and *Modeling*, on a scale from 1 to 5 (see teacher's guide). We calculated the average difficulty of each job to examine whether model accuracy varied with difficulty level. GPT ($\kappa = 0.76$) showed greater reliability on jobs with lower average difficulty (< 1.5, $n = 22$) than XGBoost ($\kappa = 0.69$). One possible explanation is that simpler jobs involve fewer player actions, which produce less variation across engineered features. This reduces the usefulness of engineered patterns that XGBoost relies on to make predictions. A similar reason also explains why

Table 1. Performances metrics for each prediction method

Model	Method	Kappa (κ)	Precision	Recall	F1
GPT	Knowledge Engineered	0.68	0.82	0.98	0.89
	Contextual Information	0.45	0.64	0.97	0.77
	Examples	0.70	0.84	0.97	0.90
XGBoost	Feature Engineering [12]	0.72	0.88	0.91	0.90

GPT ($\kappa = 0.63$) is less accurate than XGBoost ($\kappa = 0.74$) on high-difficulty jobs (avg. difficulty > 4.5, n = 15), where more complex player behavior generates patterns that are better captured by the engineered features used by XGBoost.

GPT and XGBoost show contrasting performance for two specific jobs: *Missing Whale* and *Final*. GPT matched human labels perfectly ($\kappa = 1$, n = 6), whereas XGBoost shows substantial difficulty ($\kappa < 0$). This may be due to the unusual structure of these jobs. *Missing Whale* is the only job that players can accept without the required upgrade, which often force them to quit and switch to another job before returning. *Final* consists primarily of scripted dialogue, so the player normally completes the job with minimal actions. As a result, the gameplay data for these jobs contains fewer measurable patterns that reduces the predictive power of XGBoost.

Further analysis showed that GPT failed to identify cases where long pauses between actions served as struggle indicators, possibly due to the use of cumulative timestamps in the text replays, which required interval calculations difficult for GPT [20]. We also noted that GPT was more accurate in identifying non-struggle cases (precision = 0.95) than struggle. This may stem from explicit definition of struggle, which leads GPT to label a clip as non-struggle only if it detects a complete absence of relevant indicators.

Iterative Prompt Engineering Through Review of Miscoded Cases. GPT's ability to interpret narrative elements can sometimes cause misclassifications. For example, it consistently labeled all *Eat Seaweed* job clips as struggle due to a scripted line where the character Olivia expresses uncertainty to the guiding character V1ct0r, despite the player making steady progress. To correct this, we updated the prompt to clarify the distinction between story context and player actions. The update slightly improved GPT's performance on the original 200 clips (κ: 0.70 → 0.72) and maintained stable accuracy on 100 unseen clips from round 2 ($\kappa = 0.71$), with consistent output across replications (no more than two differences across three runs on 20 clips).

2.4 Integrating GPT and XGBoost to Detect Struggle

Leveraging these findings, we created a hybrid model that combines the strengths of GPT and XGBoost for predicting struggle (see Fig. 1). If the student is working on a uniquely structured job or one with low difficulty averaged across the three dimensions, we use GPT's prediction. For jobs with high average difficulty, we follow XGBoost's prediction. In other cases where GPT predicts "Not struggle," we accept it if the clip shows no frequent pauses. If frequent pauses are present, we refer to XGBoost and assign "Not struggle" only if its predicted probability exceeds 0.75. If both models predict struggle, we label the case as such. In all remaining cases where the models disagree, we label the clip as "Not struggle" if XGBoost's probability exceeds 0.75, and "Struggle" otherwise. This threshold was selected based on the AUC-ROC curve constructed from the validation set (clips from round 2). The hybrid model achieved κ of 0.92 when validated using human-coded labels for the 200 clips from round 1. On 100 unseen clips from round 3, κ was 0.77, with precision 0.92 and recall 0.86. Although this drop suggests some overfitting, the hybrid model still outperformed the individual models ($\kappa_{XGboost} = 0.69$; $\kappa_{GPT(examples)} = 0.69$).

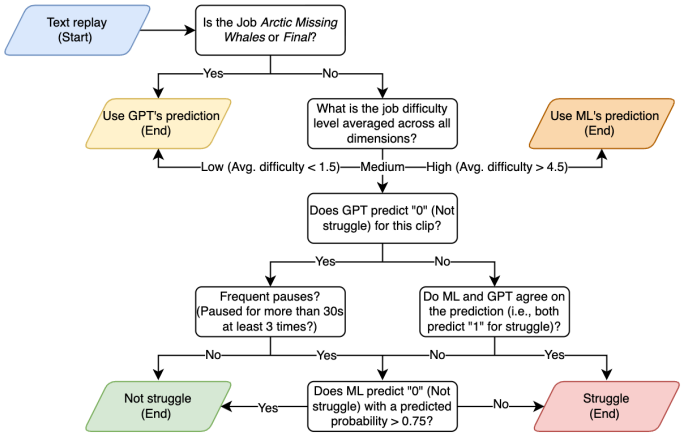

Fig. 1. Hybrid framework for predicting struggle using both GPT and XGBoost detector

3 Discussion and Conclusions

This paper examines the potential of GPT-4o to automatically detect student struggle behaviors in a science exploration game by analyzing text-based replays of log data. We find that GPT-4o performs comparably to a machine-learned detector trained for the same task but has different strengths. GPT-4o was effective in identifying struggle in simpler jobs or jobs with structures not well-represented in the training data and could use information from narrative elements or system messages. This allowed it to identify struggle that manifests in ways that are hard to quantify. GPT-4o also offers practical advantages. Building a machine-learned model requires selecting and tuning features, which can be time-consuming and technically demanding. In contrast, GPT-4o analyzes log data directly once it is converted to text. While some preprocessing is still needed, the overall setup is more accessible to researchers or educators without machine learning expertise. In contrast, XGBoost performed better when struggle followed measurable patterns, such as frequent or prolonged pauses between actions, that were well captured in the numerical features used to train the model. These differences suggest that the two approaches are complementary. Our hybrid model that combines GPT-4o and XGBoost better detected struggle in unseen data than either method alone.

This study has several limitations. First, we could not be certain that the clips labeled as instances of struggle by humans actually reflected the student's experience of struggle, a general problem in using external expert judgment to assess internal experiences. Second, GPT-4o's performance may depend on how struggle manifests in a specific game, and generalizability remains uncertain. Third, it is not entirely clear how GPT-4o arrives at its predictions, making it hard to be certain whether its judgments are based on meaningful factors or spurious correlations. Despite these limitations, GPT-4o shows promise for making behavior detection more accessible and scalable. Its ability to work directly from text replays opens new possibilities for detecting student behaviors across a range of digital learning environments.

Acknowledgments. We would like to thank NSF #DRL-1907437 for support of this project.

References

1. Anderson, J. L., & Barnett, M: Learning physics with digital game simulations in middle school science. Journal of science education and technology, 22, 914-926 (2013).
2. Baidoo-Anu, D., & Ansah, L. O.: Education in the era of generative artificial intelligence (AI): Understanding the potential benefits of ChatGPT in promoting teaching and learning. Journal of AI, 7(1), 52-62 (2023).
3. Baker, R. S., & de Carvalho, A.: Labeling student behavior faster and more precisely with text replays. In Educational Data Mining (2008).
4. Baker, R. S., Corbett, A. T., & Wagner, A. Z.: Human classification of low-fidelity replays of student actions. In Proceedings of the educational data mining workshop at the 8th international conference on intelligent tutoring systems, 29-36 (2006).
5. Barany, A., Nasiar, N., Porter, C., Zambrano, A. F., Andres, A. L., Bright, D., ... & Baker, R. S.: ChatGPT for education research: Exploring the potential of large language models for qualitative codebook development. In International conference on artificial intelligence in education,134-149 (2024).
6. Carpenter, D., Emerson, A., Mott, B. W., Saleh, A., Glazewski, K. D., Hmelo-Silver, C. E., & Lester, J. C.: Detecting off-task behavior from student dialogue in game-based collaborative learning. In Artificial Intelligence in Education: 21st International Conference, 55-66 (2020).
7. De Freitas, S.: Are games effective learning tools? A review of educational games. Journal of Educational Technology & Society, 21(2), 74-84 (2018).
8. Dicerbo, K., & Kidwai, K.: Detecting player goals from game log files. In Educational Data Mining (2013).
9. Gagnon, D. J., & Swanson, L.: Open game data: A technical infrastructure for open science with educational games. In Joint international conference on serious games, 3-19 (2023).
10. Ghali, R., Frasson, C., & Ouellet, S.: Towards real time detection of learners' need of help in serious games. In The Twenty-Ninth International Flairs Conference (2016).
11. Gobert, J. D., Baker, R. S., & Wixon, M. B.: Operationalizing and detecting disengagement within online science microworlds. Educational Psychologist, 50(1), 43-57 (2015).
12. Liu, X., Slater, S., Andres, J. M. A. L., Swanson, L., Scianna, J., Gagnon, D., & Baker, R. S.: Struggling to detect struggle in students playing a science exploration game. In Companion Proceedings of the Annual Symposium on Computer-Human Interaction in Play, 83-88 (2023).
13. Liu, X., Zhang, J., Barany, A., Pankiewicz, M., & Baker, R. S.: Assessing the potential and limits of large language models in qualitative coding. In International Conference on Quantitative Ethnography, 89-103 (2024).
14. McCall, J.: Teaching history with digital historical games: An introduction to the field and best practices. Simulation & Gaming, 47(4), 517-542 (2016).
15. Nilsson, F., & Tuvstedt, J.: GPT-4 as an automatic grader: The accuracy of grades set by GPT-4 on introductory programming assignments (2023).
16. Owen, V. E., Roy, M. H., Thai, K. P., Burnett, V., Jacobs, D., Keylor, E., & Baker, R. S.: Detecting wheel-spinning and productive persistence in educational games. International educational data mining society (2019).

17. Paquette, L., Baker, R.S.: Comparing machine learning to knowledge engineering for student behavior modelling: A case study in gaming the system. Interactive Learning Environments, 585-597 (2019).
18. Polycarpou, I., Krausea, J., Rader, C., Kembel, C., Poupore, C., & Chiu, E.: Math-City: An educational game for K-12 mathematics. Procedia-Social and Behavioral Sciences, 9, 845-850 (2010).
19. Richey, J. E., Zhang, J., Das, R., Andres-Bray, J. M., Scruggs, R., Mogessie, M., ... & McLaren, B. M.: Gaming and confrustion explain learning advantages for a math digital learning game. In International conference on artificial intelligence in education, 342-355 (2021).
20. Satpute, A., Gießing, N., Greiner-Petter, A., Schubotz, M., Teschke, O., Aizawa, A., & Gipp, B.: Can llms master math? Investigating large language models on math stack exchange. In Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval, 2316-2320 (2024).
21. Sharma, P., Stewart, A. E., Li, Q., Ravichander, K., & Walker, E.: Building learner activity models from log data using sequence mapping and hidden markov models (2024).
22. Shute, V. J., D'Mello, S., Baker, R., Cho, K., Bosch, N., Ocumpaugh, J., ... & Almeda, V.: Modeling how incoming knowledge, persistence, affective states, and in-game progress influence student learning from an educational game. Computers & Education, 86, 224-235 (2015).
23. Taub, M., Sawyer, R., Lester, J., & Azevedo, R.: The impact of contextualized emotions on self-regulated learning and scientific reasoning during learning with a game-based learning environment. International Journal of Artificial Intelligence in Education, 30, 97-120 (2020).
24. Wan, T., & Chen, Z.: Exploring generative AI assisted feedback writing for students' written responses to a physics conceptual question with prompt engineering and few-shot learning. Physical Review Physics Education Research, 20(1), 010152 (2024).
25. Yildirim, S., Narayanan, S., & Potamianos, A.: Detecting emotional state of a child in a conversational computer game. Computer Speech & Language, 25(1), 29-44 (2011).