Language Modeling for Plan Generation in Game-Base Learning Environments

Alex Goslen[1,*,†], Yeo Jin Kim[1,†], Jonathan Rowe[1] and James Lester[1]

[1]*North Carolina State University, Raleigh, NC, 27606*

Abstract

Adaptive scaffolding of students' planning activities shows significant potential for helping students regulate their learning in challenging tasks. Large language modeling provides new capabilities for augmenting such scaffolds in real-time game-based learning environments. We introduce a novel plan generation framework that leverages a text-representation of students' interactions in a game-based learning environment, Crystal Island. We formalize the plan generation task as follows: given a sequence of gameplay events, completed goals, and target goals, a language model trained on T5-small will generate a set of low-level actions to accomplish the given target goals. Gameplay data collected from 144 middle school students is used as input for the model, with 11,610 total event sequences. The generated plans are then evaluated against plans students created during gameplay through a planning support tool. The construction of this framework is designed so it will mimic a real-time system, in that it generates plans at the same point in gameplay that students do. We compare generated plans to students' plans based on how many low-level actions in the plans match. We also analyze how many actions match when mapped to high-level actions or action categories. Results show that the generated plans did largely match high-level actions with fewer low-level actions than the students' ones. This implies that the generated plan can guide the students to achieve their goals in more efficient ways. These results demonstrate potential for using language models for enhancing adaptive environments through hinting or prompting to plan efficiently in real-time.

Keywords

Goal setting and planning, Game-based learning environments, Language models, Self-regulated learning

1. Introduction

Self-regulated learning (SRL) in Winne and Hadwin's theoretical model is goal-driven learning that involves formulating goals, developing plans, and monitoring and adapting their goals and plans [1, 2, 3]. Supporting students' SRL processes can help them navigate challenging learning tasks like science problem-solving [4]. In particular, game-based learning environments have the potential to foster positive emotions through supporting *goal setting* and *planning* [5] and demonstrate the capability of adapting to individual learner strategies as students' navigate through gameplay in real-time. These learning environments generate significant amount of student data, which are essential to train machine learning models for student-adaptive learning.

AIED'23 : Empowering Education with LLMs - the Next-Gen Interface and Content Generation, July 07, 2023, Tokyo, Japan

*Corresponding author.

†These authors contributed equally.

However, typically only a fraction of the data is transformed into numbers and used by machine learning models.

With the recent developments in large language models (LLMs), there are new possibilities for data representations using trace data obtained from game-based learning environments as natural language to help support such SRL processes. Generally, LLMs are pretrained with a large corpus of text data to enable general language understanding and generation, while fine-tuning LLMs with domain-specific data allows to solve the problems important to the target domain application. This functionality creates many opportunities for educational applications, particularly in generating assistive educational content for students like practice problems, step-by-step solutions, and explanations [6]. Our work investigates the application of language models for generating plans meant to assist in students SRL processes and science problem-solving.

This work explores the use of a language model (T5) to generate potential plans for students as they navigate a science game-based learning environment, Crystal Island. A plan is defined as a series of low-level actions that can be enacted in gameplay. Textual representations of students' problem-solving activities in the game serve as input into the language model, as well as student selected goals. The language model outputs a plan, which is a series of low-level actions, that can be completed in the game. We evaluate the generated plans against plans created by students while playing the game. Our work aims to understand the effectiveness of using language models to generate plans in a game-based learning environment and discuss the implications for enhancing students' learning in real-time.

2. Related Work

Classical planning in AI involves the generation of action sequences in a given environment, which we extend to the task of plan generation in this work [7]. Techniques for plan generation have been explored for several years in the field of automated planning [8, 9, 10]. More recent approaches to plan generation involve hierarchical planning [11], and narrative plan generation with self-supervised learning [12]. These techniques can be applied to educational settings where the environment is finite and small. For example, graph networks are used to evaluate and enhance student planning in Betty's Brain [13] and have also been used for study plan generation [14]. However, game-based learning environments present a unique challenge for plan generation tasks due to the size and nature of the environment.

To generate useful plans for students in real-time, an understanding of their underlying strategies is needed. However, students' gameplay in game-based learning environments is incredibly idiosyncratic and exploratory. Recently the natural language processing (NLP) field has had much success through the development of large language models, such as tasks for summarization (BART [15]), code assisting (Copilot [16]), and dialog generation (GPT3 [17], LaMDA [18]). Our work aims to explore the performance of language models for the task of plan generation in the game-based learning environment Crystal Island.

Prior work in Crystal Island has explored the task of plan and goal recognition, which is a classification problem for predicting students' goals and plans based on their gameplay [19, 20]. Long-short term memory networks were shown to have performed best for the constructed

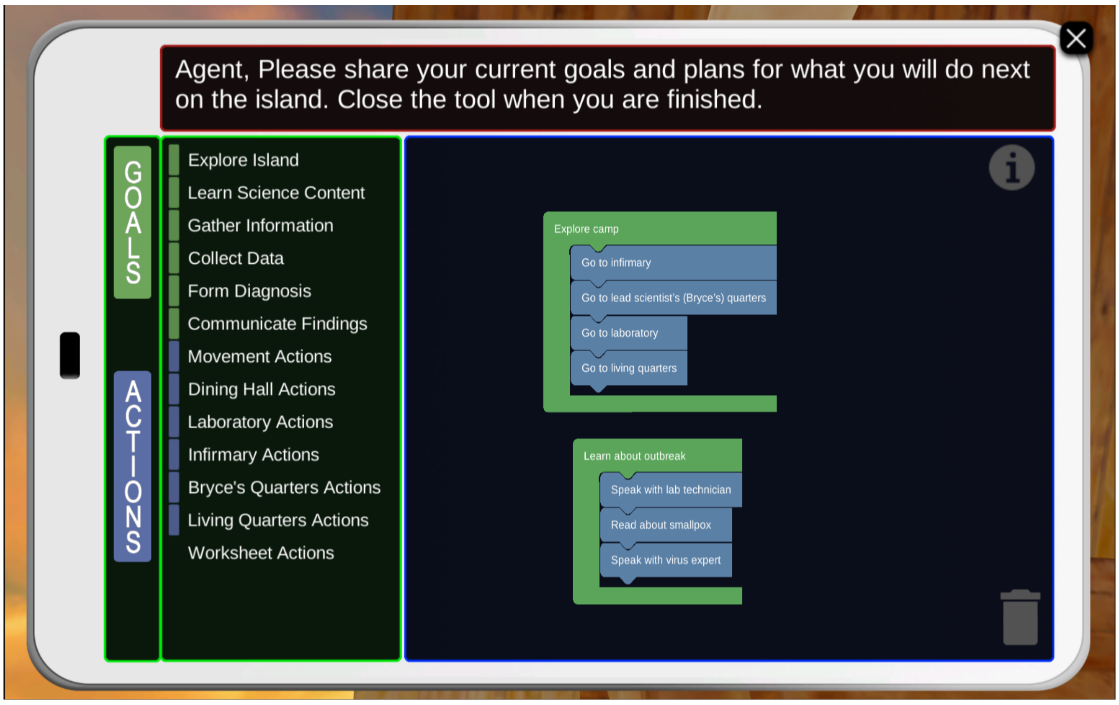
Agent, Please share your current goals and plans for what you will do next on the island. Close the tool when you are finished.

Figure 1: An example interaction with the planning support tool.

multi-task classification problem, with students' problem-solving activities being represented through one-hot encoding vectors [21, 22]. To our knowledge, language models have not been used to represent students' problem-solving activities for plan generation tasks. Through textual representations of the trace log data and textual output of plans that can be enacted in the game, language models like T5 [23] have the potential to generate more interpretable and generalizable plans.

3. Game-Based Learning Environment

3.1. Learning Environment

Crystal Island is a narrative game-based learning environment that teaches eighth grade microbiology concepts. Students are tasked with exploring a remote island to discover the source and treatment plan of a mysterious illness that has plagued the island. In this version of the game, students are presented with a planning support tool, meant to scaffold students' planning processes and help them navigate the narrative. Throughout the game, students are prompted to create plans through a drag-and-drop interface (Figure 1). In this work, we define plans to be goal clamps (green) with at least one nested low-level action (blue) inside the goal clamp. Students are able to access the planning support tool at any point in the game voluntarily as well. There were 20 goals and 55 low-level actions to choose from in the planning support tool. These goals and actions are categorized into high-level goals and actions.

3.2. Dataset

This work utilizes a dataset collected during the COVID-19 pandemic in a remote asynchronous science classroom. A total of 144 middle school students (60% female, average age 13.2 years) played the Crystal Island version containing the planning support tool over a two-day time span. Students played Crystal Island for an average of 94.7 minutes (SD = 47.7). They also completed a demographic survey, as well as pre- and post-tests. Students' problem-solving activities in the game and their planning support tool use were logged automatically as they played. The trace logs generated from these interactions were used in the analysis presented in this work and students' plans were used as a source for evaluating the generated plans.

4. Methods

In this section, we present a plan generation framework that takes in students' gameplay interactions and a goal as input. Using a fine-tuned LLM, plans are generated from the framework which consists of a series of low-level actions that can be enacted in the game. These plans are then evaluated against plans students created using the planning support tool.

4.1. Input Representation

We refer to a sequence of students' interactions with Crystal Island as an event sequence. These are logged automatically while each student is playing the game. To construct an input representation, we derived three key features from the event sequences: event type, event argument, and event location. The event type refers to 9 possible in-game activities students can complete within the learning environment. For example, reading a book and talking to a character are two types of events in the game. Event arguments are phrases that provide more detail to the event type. For example, if the event type is talking to a non-playable character, then the event argument could be the name of the character. Lastly, the event location is the location in the game where the event took place. There are 24 fine-grained locations in the game. The given example of event can be denoted as $e = (Conversation, Elise, Lab)$.

Once these three components were constructed for each event a student took, we segmented event sequences for each student according to their use of the planning support tool. Formally, we represent the input $I : [E = (e_i, e_{i+1}, ..., e_{i+k}), Completed : \mathscr{G}_c, Goals : \mathscr{G}_t]$ where E is an event sequence, consisting of a series of events, $\mathscr{G}_c$ indicates a set of completed goals, which can be an empty set, and $\mathscr{G}_t$ is the target goals the student set during planning. The aim of this framework is to generate potential plans for students, so we constructed event sequences based on the potential for real-time use. In a real-time system, the input to any model would be the action students took up to the point of prediction. A typical interaction in Crystal Island would be a sequence of gameplay events (E_1), then a planning tool interaction where they create one or more plans with a target goal ($\mathscr{G}_t = \{Explore\ camp\}$), then another series of events (E_2) and then another planning tool interaction for the next goal ($\mathscr{G}_t = \{Learn\ about\ outbreak\}$) and so on until they finish playing. A students' plan consists of a goal and a series of low-level actions. Since students interactions with the planning support tool were dispersed throughout gameplay, they serve as good points for real-time prediction of the next plan a student would make. Thus, we used the sequence of student events completed up to the point of a planning tool usage and predict possible plans at every event in the sequence.

For example, given the typical interaction with Crystal Island, then the first row of input would be $I = [E_1, \mathscr{G}_t = \{\textit{Explore camp}\}, \mathscr{G}_c = \{\}]$, since there were no completed goals. Then, in the case that a student enacted all low-level actions from plan 1 during E_2 then the next row of input would be $I = [E_2, \mathscr{G}_t = \{\textit{Learn about outbreak}\}, \mathscr{G}_c = \{\textit{Explore camp}\}]$. We constructed event sequences cumulatively with the maximum event sequence length being 30 (the median length of event sequences).

4.2. Language Model Construction

In this work, we leverage a language model to understand the student's game trace logs as text and generate the corresponding plans to achieve target goals. In the first part, the input is not a general natural language, but consists of a list of actions and location information used in the Crystal Island game domain, and thus an encoder is needed to understand the contextual representation of these characteristic inputs. In the second part, the generated plan can include actions of various lengths, and this problem is one of language generation rather than classification, so a decoder capable of language generation is required. That is, for our task, we need a Seq2Seq model with both an encoder and a decoder [24]. Various Seq2Seq language models currently exist, such as Transformer [25] and BART [15], but we used T5 for this task. T5 has the advantage of being easy to use by formatting both input and output into textual strings for various NLP tasks [23]. We trained T5-small for our plan generation models with the following hyperparameters: learning rate = 0.0003, batch size = 4, input max_length=1024, weight_decay=0.01, and warmup=1000.

4.3. Evaluation

We evaluated the language-model generated plans from the framework with the plans students created. The key evaluation metric is the percentage of actions that match between the student's plans (SP) and the generated plans (GP), with a higher percentage being better. We considered three cases for our evaluation: (1) $SP = GP:$ the case where the amount of low-level actions in the generated plans was the same as the students plan, (2) $SP < GP:$ the case where the generated plan had more low-level actions than the student plan, and (3) $SP > GP$: the case where the students' plan had more actions than the generated plan. For each of these cases, we examine how many true matches occur for each GP. A true match is when a predicted low-level action from a GP is present in the SP. For each generated plan, we counted the number of low-level actions that were a true match and divided that by the number of low-level actions in the GP, denoted as p_L. Using the high-level action mappings derived from the planning support tool, we derived a similar score based on how many high-level actions from the GP are present in the high-level SP, denoted as p_H. For example, take the "explore camp" plan in Figure 1. The low-level actions in this case are the four "go to" actions nested inside the goal. For our analysis, we would examine true matches between these and the GP low-level actions. Then we would map these low-level actions to the high-level actions and calculate p_H, which in this case would all be "explore". This analysis helps to better understand the quality of plan generation

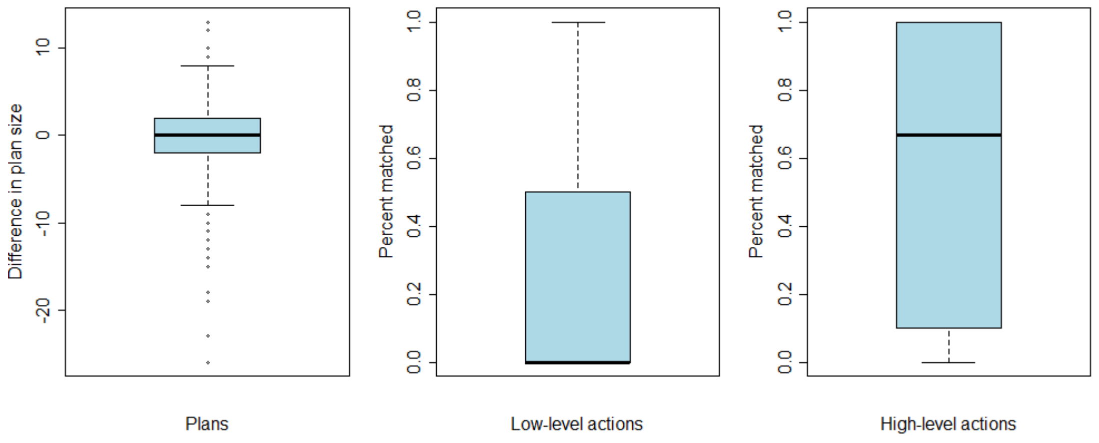

Figure 2: Distribution of difference in plan size and percent of matches across all generated plans.

and highlights potential for future work in enhancing students' planning processes, further discussed in Section 6.

5. Results

Figure 2 shows the distribution of difference in plan size between students' plans and the generated plans (left), as well as the distribution of match percentages for both low (center) and high-level actions (right). In the left boxplot, the median of the difference in plan size is 0.0 (mean=0.0, SD=3.4). A negative plan size represents the scenario where students plans were larger. The largest difference in plan size was 26, meaning students included 26 more low-level actions in their plan than the generated plan. This occurrence serves as an example of when the language models could help students build plans more efficiently. In the center boxplot, the median of match percentage in low-level actions (p_L) is 0.0% (mean=25.2%, SD=32.3%).

Out of 11,610 generated plan instances, 8.1% of these plans contained all low-level actions that were in the students' plans (exact matches) and 51.2% had no low-level actions that matched the low-level actions of the corresponding student plan. In the right boxplot, the median of match percentage in high-level actions (p_H) is 66.7% (mean=59.8% , SD=41.9%). For the high-level actions, we saw an increase in exact matches to 44.4% and a decrease in no matches to 24.8%. This indicates that while the GP did not match students' low-level actions as much, the generated plans still produced low-level actions in the same category as the student.

This is also demonstrated in Table 1 for both low-level and high-level action matching, as we see an increase in match percentage for all cases. The first case where the generated plan and the student plan were the same size (SP = GP) had the highest percentage of low-level matching. This set of generated plans could be seen as most accurate as they were the closest to students' plans. The second case with the biggest increase in match percentage was the case where the size of the students plan was larger than the generated plan (SP > GP). This indicates

Table 1
The percent of true matches between generated plans and students' plans for low-level and high-level actions.

Groups	Low-level actions (p_L)	High-level actions (p_H)
$SP = GP$	34.7%	66.0%
$SP < GP$	16.1%	46.8%
$SP > GP$	30.2%	71.2%
Overall	25.2%	59.8%

that though the students plan was more detailed, the generated plan had fewer low-level actions but in the same high-level category. This could potentially be a more efficient generated plan, which in real-time could help the student maneuver the game faster or learn more educational content. The third case where the student plan size was less than the generated plan size (SP < GP) had a much lower match percentage for both the low and high-level actions. This case needs to be explored further to better understand why the model was producing more actions and if they are viable actions for the goal. For example, a closer analysis will be needed, such as whether there is any correlation between the type of goal and the size of the plan, or whether a larger plan focuses on the effectiveness to achieve a goal rather than focusing on the efficiency.

6. Discussion

Overall, results demonstrate the promise of using language models to generate plans to prompt students' gameplay. We find that over half of the low-level actions generated by the framework align with the same high-level action category that students' used for a given plan. This indicates that even in cases where the exact matches were low, plans still contained low-level actions within the same category. Actions within the same category generally serve a similar purpose in the narrative of the game. For example, the *Explore* goal category contains primarily "go to" actions, which all serve as an exploration of the environment. Anecdotally, we observed that some unmatched low-level action plans were found in the other students' plans with the same target goals. This could be because the model generates most probable plans rather than the one personalized to a specific student. If these predictions were implemented in a real-time scenario, they could serve as more instructive hints to students or potentially more efficient plans, allowing the game to dynamically adapt to students' strategies.

A limitation of this work is that the ground truths of optimal plans are unknown or difficult to obtain. In this work, we take the students' plans as ground truth; however, we noticed a large range of planning activities. Students' use of the planning support tool varied between students for how often they opened the planning and how many plans they created throughout gameplay. While our language models seemed to generate plans similar to students', since we cannot evaluate the quality of plans it is hard to tell if the generated plans are "better" plans than what the student created. Additionally, there are several strategies that students could take in the game, which adds complexity to plan evaluation. In educational games, there are typically two overarching strategies: to win the game and to learn the curricular content. Depending on a students' primary goal, this could affect how an adaptive system would prompt students. To partially address this issue, we used an existing hierarchy to map low-level actions to high-level actions as a point for evaluation. One potential avenue for future work could be to create a larger hierarchy of actions that align with science problem solving to assess if plans follow a trajectory of scientific reasoning. This hierarchy could help evaluate both student and system generated plans.

Another interesting finding was that the language model sometimes produced low-level actions that can be played in the game, but are not present for students to select in the planning support tool. For example, students are able to pick up items in the game to test for various diseases. An output low level action was "pick up yogurt", which is an item that can be picked up in the game but it is not presented as an option for students. In the design of the planning tool, we limited the number of low-level actions presented to students so as not to overwhelm them. Thus, "pick up yogurt" was intentionally excluded, since it is not necessary to solve the mystery. However, it is interesting that the model was able to generate coherent low-level actions. This case identifies another potential area of using LLMs to generate diverse content for SRL scaffolding. The planning support tool was human designed, but considerable human effort could be reduced to have goals and actions created by LLMs trained on gameplay data. This model-based content generation could be extended to other online-based learning environments, where trace logs can have textual representations.

7. Conclusion and Future Work

This work introduces a framework that utilizes textual-representations of students' problem-solving actions within Crystal Island as input for a T5-small-based language model to generate plans that are evaluated against real-world plans created by students. Results show the potential for using such language models for providing adaptive support to students in real-time through generated plans. Currently, the framework is able to produce plans similar to plans that students created, with half of actions in generated plans aligning with the same action category as students' planned actions.

These findings highlight future work in many areas. Plan validation for science problem-solving in game-based learning environments is an open problem that requires investigation. More work needs to be done to interpret the quality of both student and system generated plans. Exploring the performance of other LLMs in this framework is another potential avenue for future work. Additionally, applying a scientific reasoning hierarchical-based approach to the language models has the potential to both help evaluate plans and enhance the generation of plans. Lastly, identifying how LLMs could enhance existing scaffolding in a real-time version of Crystal Island has the potential to enhance students' learning experiences.

Acknowledgments

This research was supported by funding from the National Science Foundation under grant DUE-1761178. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] P. Winne, A. Hadwin, Studying as self-regulated learning (pp. 291-318), 1998.

[2] P. Winne, A. Hadwin, The weave of motivation and self-regulated learning in: Schunk dh, zimmerman bj, editors. motivation and self-regulated learning: Theory, research, and application, 2008.

[3] P. H. Winne, Theorizing and researching levels of processing in self-regulated learning, British Journal of Educational Psychology 88 (2018) 9–20.

[4] D. A. Dever, M. J. Amon, H. Vrzakova, M. D. Wiedbusch, E. B. Cloude, R. Azevedo, Capturing sequences of learners' self-regulatory interactions with instructional material during game-based learning using auto-recurrence quantification analysis, Frontiers in Psychology 13 (2022).

[5] M. Boekaerts, R. Pekrun, Emotions and emotion regulation in academic settings, in: Handbook of educational psychology, Routledge, 2015, pp. 90–104.

[6] E. Kasneci, K. Seßler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier, et al., Chatgpt for good? on opportunities and challenges of large language models for education, Learning and Individual Differences 103 (2023) 102274.

[7] M. Ghallab, D. Nau, P. Traverso, Automated Planning: theory and practice, Elsevier, 2004.

[8] A. L. Blum, M. L. Furst, Fast planning through planning graph analysis, Artificial intelligence 90 (1997) 281–300.

[9] J. Hoffmann, B. Nebel, The ff planning system: Fast plan generation through heuristic search, J. Artif. Int. Res. 14 (2001) 253–302.

[10] P. Bercher, R. Alford, D. Höller, A survey on hierarchical planning-one abstract idea, many concrete realizations., in: IJCAI, 2019, pp. 6267–6275.

[11] R. Barták, S. Ondrčková, G. Behnke, P. Bercher, Correcting hierarchical plans by action deletion, in: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, volume 18, 2021, pp. 99–109.

[12] M. Polceanu, J. Porteous, A. Lindsay, M. Cavazza, Narrative Plan Generation with Self-Supervised Learning, in: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-21), , 2021, pp. 5984–5992.

[13] J. R. Segedy, J. S. Kinnebrew, G. Biswas, Using coherence analysis to characterize self-regulated learning behaviours in open-ended learning environments, Journal of Learning Analytics 2 (2015) 13–48.

[14] E. W. C. Leung, Q. Li, A dynamic conceptual network mechanism for personalized study plan generation, in: Advances in Web-Based Learning-ICWL 2003: Second International Conference, Melbourne, Australia, August 18-20, 2003. Proceedings 2, Springer, 2003, pp. 69–80.

[15] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer, BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 7871–7880. URL: https://aclanthology.org/2020.acl-main.703. doi:10.18653/v1/2020.acl-main.703.

[16] M. Chen, J. Tworek, H. Jun, et al., Evaluating large language models trained on code, CoRR abs/2107.03374 (2021). URL: https://arxiv.org/abs/2107.03374. arXiv:2107.03374.

[17] T. B. Brown, B. Mann, N. Ryder, et al., Language models are few-shot learners, CoRR abs/2005.14165 (2020). URL: https://arxiv.org/abs/2005.14165. arXiv:2005.14165.

[18] R. Thoppilan, D. D. Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H. Cheng, A. Jin, et al., Lamda: Language models for dialog applications, CoRR abs/2201.08239 (2022). URL: https://arxiv.org/abs/2201.08239. arXiv:2201.08239.

[19] W. Min, B. Mott, J. Rowe, B. Liu, J. Lester, Player Goal Recognition in Open-World Digital Games with Long Short-Term Memory Networks, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence(IJCAI-16), , New York, 2016, pp. 2590–2596.

[20] W. Min, B. Mott, J. Rowe, R. Taylor, E. Wiebe, K. Boyer, J. Lester, Multimodal goal recognition in open-world digital games, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17), 2017, pp. 80–86. doi:10.1609/aiide.v13i1.12939.

[21] A. Goslen, D. Carpenter, J. Rowe, N. Henderson, R. Azevedo, J. Lester, Leveraging Student Goal Setting for Real-Time Plan Recognition in Game-Based Learning, in: Proceedings of the Twenty-Third International Conference on Artificial Intelligence in Education (AIED-22), , 2022, pp. 78–89.

[22] A. Goslen, D. Carpenter, J. Rowe, R. Azevedo, J. Lester, Robust Player Plan Recognition in Digital Games with Multi-Task Multi-Label Learning, in: Proceedings of the 18th AAAI Conference on AIIDE , , 2022, pp. 105–112.

[23] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, JMLR 4 (2020).

[24] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, CoRR abs/1409.3215 (2014). URL: http://arxiv.org/abs/1409.3215. arXiv:1409.3215.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of the 31th Conference on Neural Information Processing SystemsNeurIPS, 2017.