

BERALL: Towards Generating Retrieval-augmented State-based Interactive Fiction Games

Rachel Chambers and Naomi Tack and Eliot Pearson
and Lara J. Martin and Francis Ferraro

{rchambel, ntack1, eliotpl, laramar, ferraro}@umbc.edu
University of Maryland, Baltimore County
Baltimore, MD, USA

Abstract

Interactive fiction (IF) games are a genre of games where the player interacts with the fictional world via text-based commands, solving puzzles primarily by exploring the world and using items they collect along the way. Although there has been much work on playing IF using AI, there is relatively less work on the creation of such games using AI. While large language models (LLMs) have made the generation of text far easier in the past several years, they still struggle to generate the highly-structured and consistent story worlds that one might see in IF. We present a three-part system called BERALL, which generates unique text adventure games by 1) maintaining the current state of the story world, 2) using retrieval-augmented generation (RAG) to create relevant location descriptions, and 3) combining these components to create a coherent experience for the player. Our approach is effective at generating room and story descriptions from the setting and knowledge graphs, demonstrating the potential benefits of LLMs in IF generation. We find challenges remain in maintaining the current game state due, in part, to LLMs not understanding the impact of changes to the knowledge graph generated by the player’s command.

1 Introduction

Interactive fiction (IF) is defined as a computer game presented and played primarily through text commands. Early and well-known forms of interactive fiction are text adventure games (TAGs). TAGs are played by presenting a description of the current location or room, with details about objects and characters in that location. Players then interact with the world by issuing commands usually in the form of “<verb> <noun>,” such as “go west” or “take apple.” The game state is maintained and updated as the player changes location, adds items to their inventory, and interacts with characters. TAGs resemble a gamified novel, ending either at

the user’s discretion or when a win condition has been reached.

Despite their seemingly simplistic text entry, TAGs have a combinatorial set of potential actions with only a small subsection of commands actually being valid (Hausknecht et al., 2019). The generation of these games is even more difficult since it is essentially generating a simulator for the player (Montfort, 2006). On top of this, TAGs generally involve puzzles or riddles for the player to solve and can easily grow to large, sprawling plots containing many characters. This creates intellectual gameplay that is hard for computer-generated text to replicate: not only must the model create puzzles but it must also keep track of a larger plot and cast of characters. TAGs can also act as controllable testbeds for coherence, plot generation, and prompting development. However, the quality of art and expression that comes with TAGs, as well as the intellectual stimulation they provide, are challenging for NLP techniques to replicate.

Most TAGs do not use automatic text generation, and all the text is written and coded by hand in bespoke languages such as Inform 7 (Nelson, 2006). This contributes to the significant amount of time and effort to write, code, and debug a single game. AI-designed games that involve language modeling could create a new and interesting experience each time. Fairly recently, we have seen interest in AI-generated IF thanks to games like AI Dungeon¹ that are built upon large language models. However, even large language models (LLMs) such as ChatGPT or LLaMA are not without flaws, as previous research into story generation (See et al., 2019; Simon, 2024) has shown how prompting with out-of-the-box models can result in text that quickly loses coherence.

We present a three-stage constrained neurosymbolic system that enforces LLMs to stay on track and generate IF with consistent game worlds. Our

¹<https://play.aidungeon.com/>

system, which we name BERALL (**BERT + RAG LLaMA**; pronounced *barrel*), has one system to "translate" the player's action into a parsed command and two LLMs to structure the game world. The first LLM is an encoder-decoder model (BERT2BERT) that is used to keep track of player actions using a knowledge graph based off of [Ammanabrolu and Riedl \(2021\)](#). The other LLM is an decoder-only model (LLaMA 2) that generates location descriptions using retrieval-augmented generation (RAG) on the knowledge graph and play history. Both LLMs work together to ensure that the generated game experience is consistent throughout the entire playthrough. We believe that our proof-of-concept system shows the potential for future work in RAG-based IF generation.

We begin the paper by highlighting previous work in TAG/IF generation and the use of knowledge-augmented LLMs for story and dialog generation. Section 3 describes the data BERALL uses and Section 4 discusses the details of its architecture. The results obtained through both ROUGE metrics and qualitative analysis are discussed in Section 5, and we end on a discussion of future directions. All code can be found on our Github².

2 Related Work

2.1 Interactive Fiction Generation

Some of the earlier work in interactive fiction generation focused on the automatic control of human-written plot points—a task known as drama management ([Nelson and Mateas, 2005](#); [Yu and Riedl, 2012](#)). These types of systems were constrained to only tell stories about whatever was pre-written. Later on, [Cychosz et al. \(2017\)](#) developed a way of using classification techniques to pick relevant IF choices from an existing dataset.

Once language models became good enough to generate the text of the game, they began to be used to generate quests ([Urbanek et al., 2019](#); [Värtinen et al., 2024](#)) or even piece together an entire game ([Ammanabrolu et al., 2020](#); [Calderwood et al., 2022](#); [Wang et al., 2024](#)).

2.2 Knowledge-Graph–Augmented Story Generation

While there have been a multitude of other ways to augment language models for story and IF generation (such as with commonsense reasoning data

([Martin, 2021](#); [Peng et al., 2022](#); [Chen et al., 2023](#)); integrating planning ([Tang et al., 2022](#); [Treyner and McCoy, 2024](#); [Simon, 2024](#)); or using other methods for retrieval ([Urbanek et al., 2019](#); [Fan et al., 2021](#))), we will focus on augmentation techniques involving knowledge graphs such as ours.

Knowledge graphs can structure data so that it is both easier for LLMs to use and easier for people to interpret. Researchers have used commonsense knowledge graphs to make more believable story continuations ([Alabdulkarim et al., 2021](#); [Chen et al., 2023](#)) as well as have generated knowledge graphs generate and keep track of quests in roleplaying games ([Ashby et al., 2023](#)). [Ammanabrolu et al. \(2020\)](#) have automatically created IF games by extracting knowledge graphs from existing stories. In our work, we use a knowledge graph to maintain the world state of the IF game and "translate" the graph into descriptions of locations, essentially implementing a version of the Parser/Describer system proposed by [Todd et al. \(2022\)](#).

Retrieval-augmented generation (RAG) ([Lewis et al., 2020](#)) is a type of neurosymbolic system that grounds the text generation of large language models with organized external information such as knowledge graphs. Although historic RAG systems finetuned pre-trained models on relevant retrieved information ([Lewis et al., 2020](#)), modern RAG uses additional corpora to *inform* a pre-trained model without finetuning. For example, [Wen et al. \(2023\)](#) have used retrieval to find similar sets of stories and then iteratively generated an "evidence forest" to fill in plot holes and rewrite the story. Like their work, our system uses a dynamic knowledge graph to prompt the LLM without finetuning.

3 Dataset

The data input into our models is taken from around 500 transcripts of Club Floyd playthroughs of TAGs³, provided directly by Jacqueline Lott ([Lott, 2023](#)), as well as the JerichoWorld dataset of knowledge graphs and room descriptions from many popular games ([Ammanabrolu and Riedl, 2021](#)). The Club Floyd transcripts were parsed, chat logs and bot commands removed, and all commands and output exported to a JSON document. This was concatenated with the JerichoWorld dataset, parsed to solely contain data about the command and the

²<https://github.com/r-chambers/TextAdventureGenerator>

³https://www.allthingsjacq.com/interactive_fiction.html

output, and given to our LLaMA 2+RAG model as the RAG documents.

3.1 About JerichoWorld

The JerichoWorld dataset contains information associated with game states, organized into n-tuples which are aggregated into a knowledge graph (Ammanabrolu and Riedl, 2021). These knowledge graphs reflect attributes of the state in a semantic way which makes them more suitable as inputs for natural language models, while providing a more computationally efficient approach. Ammanabrolu and Riedl (2021) create their knowledge graph-based prediction model using a fine-tuned sequence-to-sequence model. Their model is warm-started with BERT’s original weights as a bidirectional encoder and GPT-2’s original weights as an autoregressive decoder. The model’s input is a room description, a knowledge graph of that room, and a command. Changes to the knowledge graph after the command executes is presented to the user and used as input in the next pass through the model. Their model’s generation uses beam search to decode token by token until an end-of-sequence token is produced. Ammanabrolu and Riedl (2021) produced the JerichoWorld dataset which contains room descriptions, a knowledge graph for each description, and the knowledge graph after an action is taken.

4 BERALL System Design

BERALL consists of a user interface and two LLMs acting as a game-play state tracker and a translator between state changes and commands to room prompts. The BERT2BERT model handles these game states and the LLaMA 2 model handles the translation. We provide details of each model below. Figure 1 shows an overview of command flow.

The BERT2BERT models were trained and run on Google Colab. A High-RAM A100 GPU is needed to fine-tune and run the BERT-Base-Uncased BERT2BERT model; a T4 GPU is needed for the BERT2BERT model trained on either BERT-Medium or BERT-Tiny.

4.1 Knowledge Graph Generation

Since the model is taking in a separate context (the player’s command) in addition to generating the knowledge graph, we require our language model to have an encoder-decoder architecture.

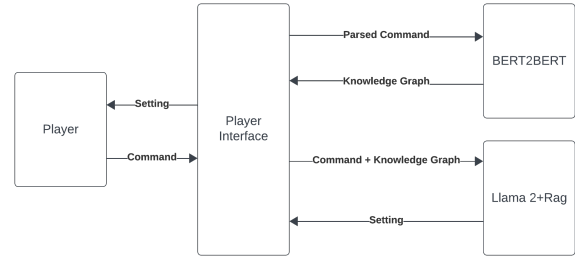


Figure 1: BERALL system architecture. Players initially receive a setting and they are able to enter any command they want. The command is parsed by the UI and sent with the current setting to the BERT2BERT state model. The BERT2BERT state model creates a new graph based on that command. The LLaMA 2+RAG model takes this graph and the game history and generates a setting based on the command.

We decided to warm-start a BERT2BERT model—using BERT for both the encoder and the decoder, following Rothe et al. (2020). When warm-starting an encoder, the architecture of the new encoder and the pre-trained checkpoint are compared, and all layers that exist in both the encoder and checkpoint are copied over and random weights are assigned to any layers that occur in the encoder but not the checkpoint. The warm-started model can then be fine-tuned on a task, saving hours in computation time while also providing better results than training a smaller model on the task directly. This model was selected due to its decent performance on translation tasks, which our knowledge graph generation task essentially is. Furthermore, warm-starting easily-accessible BERT models fit our constraints on compute power.

By having BERT as both the encoder and the decoder, it also enables the sharing of weights between encoder and decoder, which halves the model size and still achieves similar performance. Our final BERT2BERT implementation does not use this weight sharing, as BERT2BERT models that do not share weights achieved slightly better performance, as seen in Tables 1 and 2.

Our BERT2BERT model takes in the command entered and the previous knowledge graph to generate the updated graph (Figure 1: BERT2BERT). We represent the game state in a knowledge graph consisting of a list of 3-tuples. Each tuple represents information about either the current room, items in the room, items the player has, or exits from the room; for examples see Appendix B. Ammanabrolu and Riedl (2021) describe several

approaches for creating and updating knowledge graphs by describing the current state of a TAG.

Inputs to the model seen in Ammanabrolu and Riedl (2021) contained the previous room description, the previous knowledge graph, and the most recent command, outputting the new knowledge graph. However, experimentation with our model found that this input often exceeded the 512 token limit for our BERT2BERT model and created redundant output during generation. For this reason our model omits the longer room description, only taking in the knowledge graph and the command. This produces the intended output: a knowledge graph representing the state of the game after the command occurs. The BERT2BERT model was warm-started using BERT-Base-Uncased from the Transformers library⁴. Models warm-started with BERT-Medium and BERT-Tiny, respectively, were also created as baselines. All models used beam search (width = 4) rather than greedy decoding.

4.2 Setting Generation

Recent work examining LLMs and conversational games found that GPT models worked well with recall and game flow while LLaMA models became confused more often (Qiao et al., 2023). However, this approach utilized the model as a player rather than as the generator of the game. Zhu et al. (2023) found that LLaMA 2 performed better than other LLMs when generating stories, but needed more specific prompts. We hypothesize that using the model as a generator, along with providing additional context in the prompt, will help address these recall and coherence issues. Therefore, we decided to use LLaMA 2 (Touvron et al., 2023), augmented with Retrieval-Augmented Generation (RAG) (Lewis et al., 2020). Specifically, we used the 13b version of LLaMA 2, available on Hugging Face, as the base model for generating prompts (Touvron et al., 2023).

This model generates room descriptions using the knowledge graph changed by a user’s command (Figure 1: LLaMA 2+RAG). We used the collected data, as described in Section 3, as the input documentation for our LLaMA 2+RAG model. In our RAG pipeline, implemented with LangChain, the JerichoWorld and Club Floyd playthrough documents are vectorized and indexed for retrieval. We then used LangChain’s RetrievalQA library with

out-of-the-box LLaMA 2, which was given the retrieved playthrough. The LLaMA 2+RAG system is then prompted with the generated knowledge graph from the BERT2BERT model and the command from the player. By providing LLaMA 2 with context-specific prompts using RAG, the model was able to produce expected room descriptions in a similar format to human generated TAGs. We found that the LLaMA 2 model without RAG did better than expected, but the output of the LLaMA 2+RAG model was noticeably more similar in writing style to preexisting TAGs.

Scene initialization begins with a setting-knowledge graph pair, taken from JerichoWorld, presented the user. After the command is entered, the knowledge graph and command are presented to the BERT2BERT model. This model outputs a new knowledge graph changed by the command, and this graph and all previous scene descriptions and graphs are input into the LLaMA model. The next scene description is generated entirely on the fly, without further script or plot direction.

We keep the initial prompt constant for each instance of the system to produce the room description, but provide the previous knowledge graphs and model outputs as context throughout game progression as seen in Figure 2. This affords instruction, context and a few-shot base from which to generate relevant room descriptions. Several prompts were considered and those shown in Figure 2 produced the most coherent and creative descriptions. Additional prompt tuning could potentially improve the room descriptions.

4.3 User Interface

The player interface allows the user to easily understand how to give the game commands (Figure 1). We randomly select a starting graph and room description from a sub-selection of the JerichoWorld dataset to present to the player. Players then input their desired command and the game begins. As play continues the user is prompted for the next command after the model generates and displays output.

The command from the user as well as the current graph is passed to the BERT2BERT model, which then generates a new graph. The new graph is appended to the output generated by previous generations of the BERT2BERT and LLaMA 2+RAG models and passed to the LLaMA 2+RAG model to generate the next room description or set-

⁴<https://huggingface.co/blog/warm-starting-encoder-decoder>

ting. This allows both models to be stateless and maintains the entire game state in the game history.

The game does not possess a win condition or point tracking system and is more akin to a textual novel’s succession of scenes than a full game. Play only stops when the user enters the "quit" command.

5 Results

5.1 Automated Evaluation

Initial results showed that the LLaMA 2+RAG model preformed well in generating room descriptions based off the provided prompt and graphs. The outputs of this model were coherent and it provided plausible descriptions of the results of the user’s actions, most of the items, and the exits of the room. Examples can be seen in Appendix C, where realistic colloquial dialog can be seen in the model’s settings.

Initial results of the sequence-to-sequence knowledge graph generation model were less impressive, as can be seen in Appendix B. When ran after fine-tuning, all three versions of the model would create output in the correct graph format. When given knowledge graphs from JerichoWorld as inputs, the BERT2BERT model output sometimes had little to do with the input; however, when we input the new BERT2BERT generated graph into the model, the output is nearly identical to the input. Examples of knowledge graph generation can be found in Appendix B.

The recall, precision, and F1 were computed for all three iterations of the sequence-to-sequence model, shown in Tables 1 and 2. All three models showed poor performance on these measures, but improvement was generally seen as the model size increased. BERT-Base-Medium without weight-sharing had the best performance metrics overall and so the BERT2BERT model based on Bert-Base-Medium was used in the final BERALL system. BERT-Base-Uncased without weight-sharing can also be used with similar results.

5.2 Subjective Evaluation

Playing the game produced natural-sounding, appropriate output as show in Appendix C. Players going east or west was described, although the state does not always change locations, likely due to limitations with the knowledge graph generation. An example of the BERT2BERT model not always changing locations can be found as input-output

	ROUGE Precision	ROUGE Recall	ROUGE F1
BERT-Tiny	3.4	9.3	4.5
BERT-Medium	10.7	17.2	12.0
BERT-Base-Uncased	13.2	17.8	13.0

Table 1: BERT2BERT with weight sharing, evaluated on the test split of (Ammanabrolu and Riedl, 2021).

	ROUGE Precision	ROUGE Recall	ROUGE F1
BERT-Tiny	6.4	13.1	7.8
BERT-Medium	12.2	18.2	13.7
BERT-Base-Uncased	10.3	18.2	11.6

Table 2: BERT2BERT without weight sharing, evaluated on the test split of (Ammanabrolu and Riedl, 2021).

pair B.4 in Appendix B. However, the LLaMA 2+RAG model responded to commands to examine items, go directions, and talk to characters with appropriate responses. Occasionally, option menus for the user were printed despite the prompt specifying not to print options. Regardless, the options were interesting and relevant to the setting.

6 Discussion & Future Work

Data preparation presented an obstacle, as implied exits had to be added to the knowledge graphs. Additionally, exits parsed from the JerichoWorld dataset had to account for multiple ways of moving between locations in a game.

Prompt tuning has been shown to help models perform better in zero- and few-shot learning environments (Lester et al., 2021; Li and Liang, 2021; Hu et al., 2022; Chen et al., 2022; Ajwani et al., 2024), so further considerations on how to phrase prompts for TAGs is recommended.

Future work could explore using the LLaMA 2+RAG model to start the TAG using a larger selection of games. This would allow the player to choose what kind of adventure they would like to have and highlight the ability of the LLM to generate games from many different starting points.

Longer-term goals could consider supplementing the RAG documents with additional interactive stories, potentially utilizing *Choose Your Own Adventure*-esque books to give the model further creative works to draw from. This would involve dataset curation and significant parsing to produce the related graphs. Another consideration could be fine-tuning the BERT2BERT model to generate knowledge graphs that are more closely related to the input. Altering the fine-tuning with time-series

commands and updated graphs might prove to be effective.

As mentioned above, the prompt provided to the BERT2BERT model can also be expanded and improved via few-shot examples. This may produce an output more related to the prompt but may add to the struggle of identifying object relationships. Considering the strategies used in models which predict the next actions given a video input might aid understanding relationships between objects and places. Given these models' more complete grasp of an object-place relationship, they might lend themselves to accurate generation of a knowledge graph.

7 Conclusion

We present our system, BERALL, which adapts LLMs to generate TAGs. We utilize a user interface to handle connecting the knowledge graph model, BERT2BERT, and the room description model, LLaMA 2+RAG, to progress the game. LLMs show promising results for generating playable TAGs, but also indicate room for improvement. LLMs prove to be challenged when asked to make changes to a game state. This initial development shows great potential for generating room descriptions with minimal adjusting of existing LLMs. This is promising as it reduces the environmental impact of adapting models to the storytelling domain. BERALL highlights the power and limitations of LLMs when it comes to generating interactive fiction.

Acknowledgements

We would like to thank our classmate Patricia Delafuente for her advice on using the LLaMA 2+RAG model. We would also like to thank Jacqueline Lott and Club Floyd for their many transcripts, without which our RAG model would not have had such impressive results.

References

Rohan Deepak Ajwani, Zining Zhu, Jonathan Rose, and Frank Rudzicz. 2024. [Plug and Play with Prompts: A Prompt Tuning Approach for Controlling Text Generation](#). In *Deployable AI Workshop at AAAI-2024*.

Amal Alabdulkarim, Winston Li, Lara J. Martin, and Mark O. Riedl. 2021. [Goal-Directed Story Generation: Augmenting Generative Language Models with Reinforcement Learning](#). *arXiv preprint arXiv:2112.08593*.

Prithviraj Ammanabrolu, Wesley Cheung, Dan Tu, William Broniec, and Mark O. Riedl. 2020. [Bringing Stories Alive: Generating Interactive Fiction Worlds](#). *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 16(1):3–9.

Prithviraj Ammanabrolu and Mark Riedl. 2021. [Modeling worlds in text](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Trevor Ashby, Braden Webb, Gregory Knapp, Jackson Searle, and Nancy Fulda. 2023. [Personalized Quest and Dialogue Generation in Role-Playing Games: A Knowledge Graph- and Language Model-based Approach](#). In *ACM Conference on Human Factors in Computing Systems (CHI)*, pages 290:1–290:20.

Alex Calderwood, Noah Wardrip-Fruin, and Michael Mateas. 2022. [Spinning Coherent Interactive Fiction through Foundation Model Prompts](#). In *International Conference on Computational Creativity (ICCC)*.

Jiaju Chen, Yuxuan Lu, Shao Zhang, Bingsheng Yao, Yuanzhe Dong, Ying Xu, Yunyao Li, Qianwen Wang, Dakuo Wang, and Yuling Sun. 2023. [FairytaleCQA: Integrating a Commonsense Knowledge Graph into Children's Storybook Narratives](#). *arXiv preprint arXiv:2311.09756*.

Yulong Chen, Yang Liu, Li Dong, Shuhang Wang, Chenguang Zhu, Michael Zeng, and Yue Zhang. 2022. [AdaPrompt: Adaptive model training for prompt-based NLP](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6057–6068, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Margaret Cychosz, Andrew S. Gordon, Obiageli Odimegwu, Olivia Connolly, Jenna Bellassai, and Melissa Roemmele. 2017. [Effective Scenario Designs for Free-Text Interactive Fiction](#). In *International Conference on Interactive Digital Storytelling (ICIDS)*, volume 10690 of *LNCS*, page 12–23. Springer International Publishing.

Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2021. [Augmenting Transformers with KNN-Based Composite Memory for Dialogue](#). *Transactions of the Association for Computational Linguistics (TACL)*, 9:82–99.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2019. [Interactive Fiction Games: A Colossal Adventure](#). *AAAI Conference on Artificial Intelligence (AAAI)*, 34(4):7903–7910.

Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Jingang Wang, Juanzi Li, Wei Wu, and Maosong Sun. 2022. [Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2225–2240.

- Dublin, Ireland. Association for Computational Linguistics.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. [The power of scale for parameter-efficient prompt tuning](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks](#). In *Advances in Neural Information Processing Systems*, volume 33, page 9459–9474, Vancouver, Canada. Curran Associates, Inc.
- Xiang Lisa Li and Percy Liang. 2021. [Prefix-tuning: Optimizing continuous prompts for generation](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.
- Jacqueline Lott. 2023. Personal Communication.
- Lara J. Martin. 2021. *Neurosymbolic Automated Story Generation*. Phd, Georgia Institute of Technology, Atlanta, GA.
- Nick Montfort. 2006. [Natural Language Generation and Narrative Variation in Interactive Fiction](#). In *AAAI Conference on Artificial Intelligence (AAAI)*, page 45–52.
- Graham Nelson. 2006. *The IF Theory Reader*, chapter Natural Language, Semantic Analysis, and Interactive Fiction.
- Mark J. Nelson and Michael Mateas. 2005. [Search-Based Drama Management in the Interactive Fiction Anchorhead](#). *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 1(1):99–104.
- Xiangyu Peng, Kaige Xie, Amal Alabdulkarim, Harshith Kayam, Samihan Dani, and Mark Riedl. 2022. [Guiding neural story generation with reader models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7087–7111, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Dan Qiao, Chenfei Wu, Yaobo Liang, Juntao Li, and Nan Duan. 2023. [GameEval: Evaluating LLMs on Conversational Games](#). *arXiv preprint arXiv:2308.10032*.
- Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. [Leveraging pre-trained checkpoints for sequence generation tasks](#). *Transactions of the Association for Computational Linguistics*, 8:264–280.
- Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D. Manning. 2019. [Do Massively Pretrained Language Models Make Better Storytellers?](#) In *Conference on Computational Natural Language Learning (CoNLL)*, page 843–861, Hong Kong, China. Association for Computational Linguistics.
- Nisha Simon. 2024. [Large Language Models are Incoherent Storytellers](#). In *Canadian Conference on Artificial Intelligence (CAIAC)*, Online. Canadian Artificial Intelligence Association.
- Chen Tang, Zhihao Zhang, Tyler Loakman, Chenghua Lin, and Frank Guerin. 2022. [NGEP: A graph-based event planning framework for story generation](#). In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 186–193, Online only. Association for Computational Linguistics.
- Graham Todd, Zegang Cheng, Yifan Liu, and Julian Togelius. 2022. [Towards Knowledge-Graph Constrained Generation for Text Adventure Games](#). In *Wordplay Workshop at NAACL 2022*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Auralien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. [Llama 2: Open Foundation and Fine-Tuned Chat Models](#). *arXiv preprint arXiv:2307.09288*.
- Nicholas Sloss Treynor and Joshua McCoy. 2024. [College Ruled: A Pathfinding Approach to Generative Storytelling](#). In *International Conference on the Foundations of Digital Games (FDG)*, FDG '24, page 15:1–10, Worcester, MA, USA. Association for Computing Machinery.

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. 2019. [Learning to Speak and Act in a Fantasy Text Adventure Game](#). In *Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, page 673–683, Hong Kong, China. Association for Computational Linguistics.

Susanna Värtinen, Perttu Hämäläinen, and Christian Guckelsberger. 2024. [Generating Role-Playing Game Quests With GPT Language Models](#). *IEEE Transactions on Games*, 16(1):127–139.

Ruoyao Wang, Graham Todd, Ziang Xiao, Xingdi Yuan, Marc-Alexandre Côté, Peter Clark, and Peter Jansen. 2024. [Can Language Models Serve as Text-Based World Simulators?](#) In *Annual Meeting of the Association for Computational Linguistics (ACL)*, Bangkok, Thailand. Association for Computational Linguistics.

Zhihua Wen, Zhiliang Tian, Wei Wu, Yuxin Yang, Yanqi Shi, Zhen Huang, and Dongsheng Li. 2023. [GROVE: A Retrieval-augmented Complex Story Generation Framework with A Forest of Evidence](#). In *Findings of the Association for Computational Linguistics: EMNLP*, page 3980–3998, Singapore. Association for Computational Linguistics.

Hong Yu and Mark O. Riedl. 2012. [A Sequential Recommendation Approach for Interactive Personalized Story Generation](#). In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 1, page 71–78.

Hanlin Zhu, Andrew Cohen, Danqing Wang, Kevin Yang, Xiaomeng Yang, Jiantao Jiao, and Yuandong Tian. 2023. [End-to-end story plot generator](#). *arXiv preprint arXiv:2310.08796*.

A Prompt for Setting Generation

Below, in Figure 2, is the prompt that was used for generating the room descriptions using LLaMA 2.


```

<s>[INST] <<SYS>>
Convert graphs into the output of a text adventure game, like Zork.
Be artful and expressive.
Always use second person.
Do not give the user any choices about what to do next.
Do not tell the user how to enter the next command.

The first part of the graph will be the command in the format
  'command', 'is', '{{ command }}'.
For the 'use {{ item }}' command, describe the user using the item.
For the 'take {{ item }}' command, describe the user taking the item.
For the 'talk to {{ person }}' command, describe the user talking to the
  person and the conversation they have.
The graph will have 'you', 'in', '{{ room name }}' representing the room
  the player is in.
The graph will also have items in the room, represented by
  '{{ item name }}', 'in', '{{ room name }}'.

The player will also have inventory items, represented by
  'you', 'have', '{{ item name }}'.
Do not mention the inventory items in the text.
If the player does not have an item in their inventory, then they can't
  use that item via the 'use {{ item }}' command.

Finally, exits to the current room will be in the format
  '{{ room name }}', 'is', '{{ direction }}'.
Mention all the exits of a room and their directions.

Here are examples of converting graphs to output:
((Graph and Output pairs, omitted for brevity)) <</SYS>>
{{ [['command', 'is', 'examine map'],
['you', 'in', 'Captain's Cabin'],
['map', 'in', 'Captain's Cabin'],
['Captain's chair', 'in', 'Captain's Cabin'],
['painting', 'in', 'Captain's cabin'],
['desk', 'in', 'Captain's cabin'],
['Deck', 'is', 'east']] }} [/INST] </s>
<s> [INST]
((Previous Graph + Output pairs are appended here after
generation))
[/INST] </s>

```

Figure 2: Prompt provided to the LLaMA 2 model for room description generation. Template variables are in curly brackets. Line breaks and red coloring for template variables were added for readability in the paper.

B Example Graphs

B.1

input: [['command', 'is', 'east'], ['you', 'have', 'piece of white paper'], ['you', 'in', 'Closet'], ['small black pistol', 'in',

'Closet'], ['Chief's office', 'is', 'east']]
output: [['you ', 'have ', 'flashlight'], ['you ', 'in ', 'hall'], ['hall ', 'is ', 'south'], ['hall outside elevator ', 'is ', 'east']]

B.2

input: [['command', 'is', 'east'], ['Behind House', 'is', 'east'], ['Forest Path', 'is', 'north'], ['West House', 'is', 'west']]

output: [['grass', 'in', 'forest pathway'], ['empty stone bird bath', 'in', 'forest pathway'], ['you', 'have', 'large iron key'], ['you', 'have', 'pewter mug'], ['trees', 'in', 'forest pathway'], ['birds', 'in', 'forest pathway'], ['exits', 'in', 'forest pathway'], ['you', 'have', 'bunch of bananas'], ['ferns', 'in', 'forest pathway'], ['sky', 'in', 'forest pathway'], ['you', 'in', 'forest pathway'], ['forest pathway', 'is', 'west'], ['post office', 'is', 'east']]

B.3

input: [['command', 'is', 'south'], ['you', 'in', 'Castle'], ['jet black sword', 'in', 'The Dark Elf'], ['The Dark Elf', 'in', 'Castle'], ['you', 'have', 'fine Pentarian sword'], ['Castle Entrance', 'is', 'south'], ['Main Hall', 'is', 'north'], ['Sitting Room', 'is', 'east']]

output: [['you', 'in', 'forbidding castle'], ['you', 'have', 'large cardboard box'], ['castle door', 'in', 'forbidding castle'], ['walls', 'in', 'forbidding castle'], ['scrub', 'in', 'forbidding castle'], ['you', 'have', 'small screwdriver'], ['you', 'have', 'large iron key'], ['you', 'have', 'pewter mug'], ['exits', 'in', 'forbidding castle'], ['you', 'have', 'bunch of bananas'], ['you', 'have', 'small silver bell'], ['castle', 'in', 'forbidding castle'], ['sky', 'in', 'forbidding castle']]

B.4

input: [['command', 'is', 'west'], ['you', 'have', 'sword'], ['you', 'in', 'shadow armory'], ['light', 'in', 'shadow armory'], ['stones', 'in', 'shadow armory'], ['you', 'have', 'bas - relief'], ['shadow weapons', 'in', 'shadow armory'], ['stairs', 'in', 'shadow armory'], ['you', 'have', 'armor and silks'], ['bas - relief', 'in', 'shadow armory'], ['stairs', 'in', 'shadow armory'], ['undying garden', 'is', 'north'], ['silent symphony', 'is', 'west'], ['moonlit chamber', 'is', 'up']]

output: [['you', 'have', 'armor and silks'], ['you', 'in', 'shadow armory'], ['light', 'in', 'shadow armory'], ['stones', 'in', 'shadow armory'], ['you', 'have', 'porcelain mask'], ['shadow weapons', 'in', 'shadow armory'], ['tasseled pommel', 'in', 'bronze - hilted dagger'], ['white tassel', 'in', 'bronze - hilted dagger'], ['bas

- relief', 'in', 'shadow armory'], ['stairs', 'in', 'shadow armory'], ['undying garden', 'is', 'north'], ['silent symphony', 'is', 'west'], ['moonlit chamber', 'is', 'up']]

C Example Playthrough

NIGHT AT THE COMPUTER CENTER

An Interactive Nocturnal Chase

Copyright (c) 1996 by bonni mierzewska.

Release 1 / Serial number 961118 / Inform v1502
Library 5/12

Standard interpreter 1.0

Computer Site: This is the computer site. You know, half a dozen PCs, couple of Macs, a printer table, the desk at which you write that interactive fiction game instead of study. An internet router hums quietly in the corner.

The door is northeast.

[The output generated by the model starts here. Text in bold are commands input by the player.]

Please enter a command: examine pcs

You find yourself in a computer site filled with various machines and gadgets. A desk with a mouse and a mac lies nearby, while a sticker on the monitor catches your eye. You notice a printer table with a laser printer and a stack of papers. A Uggs manual rests on the desk, and you spot a compile game icon on the computer. The computer itself sits on a desk, surrounded by icons for lpr, telnet, ftp, and internet routers.

What would you like to do?

Please enter a command: east

You decide to head east. As you move through the computer site, you notice a group of technicians working on a large machine. They seem to be having trouble with the device, and they look frustrated. One of them notices you and approaches you.

"Hey, you look like someone we can trust. We're trying to fix this internet router, but it's not cooperating. Do you think you could help us out?"

What would you like to do?

Please enter a command: help them out

You approach the group of technicians and ask if you can assist them. One of them explains that the internet router has been acting strangely, causing connectivity issues throughout the computer site. They have tried various troubleshooting methods, but nothing seems to work.

You examine the router and notice that one of its LED indicators is flashing rapidly. After some

thought, you suggest trying to reset the router to its default settings. The technicians agree that this is worth a try and thank you for your help.

As you watch, they successfully reset the router, and the LED indicator stops flashing. The technicians breathe a collective sigh of relief and begin testing the internet connection.

With the issue resolved, the technicians return to their workstations, and you continue exploring the computer site.

What would you like to do next?

Please enter a command: examine internet router

You take a closer look at the internet router and notice that it is a high-end model with advanced features and settings. You see several LED indicators on the front panel, each indicating different aspects of the router's performance.

One of the LEDs is flashing rapidly, which suggests that there might be an issue with the router's network connection. You wonder if this could be related to the connectivity problems the technicians were experiencing earlier.

As you continue to examine the router, you notice a small button on the side panel labeled "Reset." You press the button, and the router's LED indicators start blinking in a pattern that indicates it is resetting to its default settings.

After a few moments, the router completes the reset process and returns to its normal operation. You notice that the LED indicator that was flashing before is now steady, indicating that the issue has been resolved.

What would you like to do next?