Extended Abstract

Motivation Prior work on using LLMs to play text-based adventure games has identified several limitations, namely that state-of-the-art models struggle to use knowledge they have collected or identify intermediate goals on the path toward completing the game. More specifically, GPT-4 achieved an accuracy of 75% on one-step questions, or questions about possible moves that could directly follow the current move, and often defined goals only in relation to its current position from Tsai et al. (2025). This suggests that the LLM's "working memory" is insufficient to keep track of objectives in an open world, and this lack of memory also keeps its goals short sighted. Reinforcement learning offers opportunities to give clear reward signals for exploration and encourage decisions that maximize the potential for rewards several steps in the future.

Text games have several features that make them an ideal task to test the limits of reinforcement learning techniques. Primarily, there is no clear human "baseline" to benchmark against or imitate (although we try to create one for some way of comparison). When combining unclear reward signals with nebulous project definitions, we uncover a rich suite of environments with which to test meta-reinforcement learning (meta-RL) approaches.

Method Our model and approach is most directly based on Li et al. (2023) and their potential based reward shaping method. At its core, our model is a soft actor-critic (SAC) model that takes in a state vector, generates an action in the form of a word embedding, then selects the available action that has the minimal euclidean distance to our generated action vector. We use a LLAMA 3.2B Instruct model to generate the embeddings of states and actions.

Our novel contribution and the focus of our project is the use of different reward shaping techniques. Inspired by Li et al. (2023) and their use of the current estimate of the value function as a potential function for reward shaping, we use several approaches to help us find reward signals in addition to the Jericho environment's sparse rewards.

We trained six models, trying combinations of reward shaping methods (potential based and learned auxiliary net) and curriculum learning methods (random state initialization, walking back from the end, walking forward from the beginning).

Implementation We have implemented the entire SAC framework and training loop. We also translated the Microsoft Jericho framework into a gym-esque framework to integrate into a training loop. We wrote both an offline and online version of our training framework. Our implementation is linked here: https://github.com/pabaill/text-game-rl.

Results Our best model used potential based reward shaping and a curriculum learning approach that prioritizes later states. This model performed on par with the current version of Chat GPT on the same game, albeit requiring more iterations to do so. This version of our model balanced exploration and exploitation better than any other model (0.45/0.55 split, versus 0.2/0.8 from other models), and we believe this model benefited from using critic pretraining to better calibrate the potential based reward shaping. Qualitatively, potential based methods without proper critic pretraining tended to interact with immediate objects and not explore the environment well, and learned reward nets, regardless of the curriculum learning method, also ran into this issue.

Discussion Even with nucleus sampling, our models often struggle with repeating the same, high likelihood action several times. By increasing the choices from which the model can select, we risk choosing actions that are entirely irrelevant. As such, future work could consider methods of augmenting action selection by penalizing repeated actions or using other post processing steps to enhance model output.

Conclusion We believe there is significant potential to using potential based reward shaping (given proper critic pretraining) to solve open ended problems without an LLM alone. We hope that future work considers applications outside of text-based games and continues to explore the effectiveness of reward shaping in sparse reward environments.

Logits and Labyrinths: Using Meta RL to Play Text Based Adventure Games

Javokhir Arifov
Department of Linguistics
Stanford University
javokhir@stanford.edu

Philip Baillargeon
Department of Computer Science
Stanford University
pabaill@stanford.edu

Nathanael Cadicamo
Department of Symbolic Systems
Stanford University
cadicamo@stanford.edu

Abstract

Text-based adventure games present a challenging, open-ended problem solving environment for testing learning methods. These games feature several intermediary challenges with sparse rewards and often unclear objectives. Even with the help of experts, humans struggle with these games. While there were hopes that LLMs would be able to navigate these games (Tsai et al. (2025)), their lack of working memory and small context window make this difficult.

We propose a soft actor critic model (SAC) that builds upon Li et al. (2023) to integrate curriculum learning and reward shaping while re-implementing a version of their model from scratch as a baseline. Our training framework with an explicitly learned reward shaper produces a lightweight, specialized, general purpose problem solving model for text-based adventure games with the potential to outperform LLMs alone. We identify reward shaping and reward buffer curation as two critical components of model architecture to consider when creating future agents.

1 Introduction

Researchers in reinforcement learning (RL) have been using text-based adventure games as a benchmarking opportunity for open-ended task completion Osborne et al. (2022). With the advent of LLMs, research into this area has slowed, but there are some signs that the training of lightweight general purpose models for text game playing might become an increasing focus of research.

Text games have several features that make them an ideal task on which to test the limits of reinforcement learning techniques. Primarily, there is no clear human "baseline" to benchmark against or imitate. Text-based adventure games are difficult for even the most experienced of humans because they represent open-ended series of problems across space and with unpredictable interactions between objects in an environment. They also tend to be nonlinear, making the assessment of progress difficult. When combining unclear reward signals with nebulous project definitions, we uncover a rich suite of environments with which to test meta-reinforcement learning (meta-RL) approaches. More specifically, we use this project to explore methods of shaping rewards to modify these sparse signals and methods of introducing new data to promote learning (curriculum learning from Bengio et al. (2009)).

We train several agents to play Zork, a text-based adventure game developed at MIT in 1977 (Lebling et al. (1979)). One of the first "computerized fantasy games", Zork is a massive world filled with

trolls, mazes, and other assorted puzzles, all controlled by a player through natural language text commands. At the time, this system was notable for its ability to parse open-ended text. Today, it represents a complex training ground for text-based agents. We use reinforcement learning to attempt to beat current LLM benchmarks for Zork as an experiment in open-ended, text-based problem solving.

2 Related Work

Prior work on using LLMs to play text-based adventure games has identified several limitations, namely that state-of-the-art models struggle to use knowledge they have collected or identify intermediate goals on the path toward completing the game. More specifically, GPT-4 achieved an accuracy of 75% on one-step questions, or questions about possible moves that could directly follow the current move, and often defined goals only in relation to its current position from Tsai et al. (2025). This suggests that the LLM's "working memory" is insufficient to keep track of objectives in an open world, and this lack of memory also keeps its goals shortsighted. Reinforcement learning offers opportunities to give clear reward signals for exploration and encourage decisions that maximize the potential for rewards several steps in the future.

Deep learning on knowledge graphs has been a frequent subject of prior research such as Ammanabrolu and Riedl (2019); Zelinka et al. (2020). This is because text-based adventure games can be abstracted into a graph in which the nodes are individual rooms, the edges are actions, and the agent must learn the correct order to traverse these vertices in order to achieve the highest score. However, this approach seems unlikely to yield knowledge that will generalize to a world model, as this would require a potentially intractably large knowledge graph to traverse for a general-purpose model. As such, we focus our project on language modeling and efficient mechanisms to choose a correct action with minimal context as found in Yao et al. (2020); Ryu et al. (2023).

Since this task requires a mix of exploration and exploitation, actor-critic methods are common. Li and colleagues use a soft actor-critic (SAC) approach from Li et al. (2023), in which the agent seeks to maximize rewards and entropy to encourage exploration, multiple critics to prevent overestimation, and a replay buffer similar to PPO. Their system was quite successful, but since they were using hashed actions fed into a deep neural network that lacked additional context, they often found their model was stuck in loops of similar actions that did not progress the game. We believe, as the authors suggest, that an LLM with additional context could circumvent these shortcomings.

In completing our project, we came to appreciate the importance of intentionally introducing salient information. The subset of meta-RL, curriculum learning by Bengio et al. (2009), emphasizes the importance of using salient examples for developing optimal game-playing strategies. In chess, this could be board states with clear optimal moves, the potential for checkmates (a clear negative reward signal), or near end states with short horizons. These principles have clear analogues in our case that we hope to exploit.

3 Method

Our model and approach are inspired by Li et al. (2023) and their potential-based reward shaping method. At its core, our model is a soft actor-critic (SAC) model that takes in a state vector, generates an action in the form of a word embedding, and then selects the available action that has the minimal euclidean distance to our generated action vector. We use a LLAMA 3.2B Instruct model to generate the embeddings of states and actions. State embeddings include the last text description of the setting and the player's inventory.

Our novel contribution and the focus of our project is the use of different reward shaping techniques. Inspired by Li et al. (2023) and their use of the current estimate of the value function as a potential function for reward shaping, we use numerous approaches to help us find reward signals in addition to the Jericho environment's sparse rewards. We also experiment with other reward shaping techniques that have the potential to better develop general strategies for finding clearer reward signals.

3.1 Potential Based Reward Shaping

Potential based reward shaping modifies our original reward equation with an additional term, F, and potential function, Φ, where:

$$Q_{\text{new}}(s,a) = Q_{\text{old}}(s,a) + \alpha[r + F(s,s') + \gamma \max_{a'} Q_{\text{old}}(s',a') - Q_{\text{old}}(s,a)] \tag{1}$$

$$F(s,s') = \gamma\Phi(s') - \Phi(s) \tag{2}$$

In Li et al. (2023), they use the value function $\Phi(s) = V_{\text{approx}}(s)$, or the estimation of the value function at that given episode, as a potential function. The motivation behind this method is to reward progression to "high value" states, as these will likely lead to real, high rewards in the future.

3.2 Learned Reward Shaping

We also experimented with a learned reward shaping network $S(s,s')$ which learns the potential difference between the two functions. Our reward then becomes:

$$r_{\text{shaped}} = r + \lambda S(s,s') \tag{3}$$

And the loss of our shaper S is:

$$\delta_t = (r + \gamma(1-d) \cdot \text{critic}(s', \text{actor}(s')) - \text{critic}(s,a))$$
$$\mathcal{L}_S = \text{MSELoss}\left(\text{reward\_delta}, \delta_t\right)$$

Rather than being sensitive to point estimates of the value function, this method explicitly learns to predict rewards that minimize TD error, and thus entice the model into moving toward intermediate states between rewards. We also use some optimization features such as gradient clipping on the reward shaper network to prevent the reward signal from exploding and giving rewards that are disproportionate to their respective actions.

3.3 Curriculum Learning

Another essential component of our approach is the method through which we show our models experiences. We use curriculum learning (Bengio et al. (2009)) to conceptualize what experiences may be the most valuable to our model. We first consider initializing our model to a random state in the walkthrough at each training step, giving it varied experiences while maintaining that forward progression through the game is still possible. We also consider initializing the model at the final state of the game, then slowly walking it backwards as it gathers rewards at the game's final steps. The motivation behind this approach is to give the agent experiences with high reward signals early in training, which are most often found at the end of the game. We finally try initializing the agent at the beginning, then stepping the starting point forward as training progresses. The beginning of these games are often open-ended, and we hope that this approach would give our agent diverse experiences to learn from.

4 Experimental Setup

4.1 Offline Approach

Initially, we generated a dataset of 30,000 (state, action, next_state, reward, done) tuples to train our agent without having to query the game. We did this by using Jericho's walkthroughs of the games, interspersing some random actions intermittently. This gave us a representative mix of reward gathering and exploration actions to train our agent. This had the benefit of allowing us to precompute all embeddings before the training loop, thus focusing training time on action generation rather than encoding novel states.

We quickly abandoned this approach because it significantly limited the actions we could see and the ability of the model to build memory. Because these tuples lacked context, our trained models prioritized exploration steps (e.g., "west", "pick up map") that did not progress the agent toward the goal or effectively use the items in its inventory.

Table 1: Performance Comparison for Zork

Reward Shaper	Buffer Style	Explore/Exploit	Final Score	Episode Length
Human	Human	-	74	173
Chat GPT	Chat GPT	-	35	**34**
Potential Based	Random	0.18/0.82	0	1000
Potential Based	Curriculum (beginning)	0.18/0.82	0	1000
Potential Based	Curriculum (end)*	**0.45/0.55**	**35**	1000
Learned	Random	0.38/0.62	0	1000
Learned	Curriculum (beginning)	0.28/0.72	0	1000
Learned	Curriculum (end)*	0.38/0.62	0	1000

*** critic pretrained for ten steps to ensure quality critic before actor steps**

4.2 Online Approach

For our online approach, we required heavy optimization in creating a gym environment given the slow game files. As such, we calculate and cache each computed action embedding a single time to avoid expensive calls to the LLAMA embedder. Additionally, we use curriculum learning (Bengio et al. (2009)) to vary our selection of random, end state, or beginning state experiences. Our implementation of this environment and training loop are linked in the appendix.

Each of our six models were trained for 100 episodes, which took 4-6 hours on average for each model. All models were trained on Zork 1 using a custom wrapper that managed the state in the game environment. We then use an evaluation environment and calculate general statistics about the action selections as well as saving the first 1000 actions taken by the model.

5 Results

5.1 Quantitative Evaluation

For quantitative results, we begin with summaries of model performance for each of our models in Table 1. The "explore/exploit" column represents the ratio of exploration actions (e.g., movement like "north" or "south") and exploitation actions (e.g., object manipulation like "pick up sword" or "move rug"). The learning curves for our best model are in Figures 1, 2, and 3.

Of particular note with respect to our human baseline, a walkthrough was used for assistance as three "lifelines". Without the walkthrough, we would not have been able to play the game to its first major checkpoint, a sufficient benchmark for our purposes. As a three-person team, we ourselves struggled with finding the first reward-bearing action (e.g. finding an entrance to the house), which did not bode well for our trained models. As expected, many of them could not find this first step either.

Unsurprisingly, our models that used randomly initialized states performed very poorly. These models tended to use objects in their immediate surroundings and failed to properly explore their environment. As such, they rarely proceeded past the first "room" of the game, the front of the house, where there is nothing to interact with except a mailbox.

Our learned reward shaping network performed poorly across all curriculum learning methods. We suspect this is because our shaped reward quickly plateaued as the model was unable to escape initial states, and was therefore prone to the same pitfalls as a model without reward shaping. Thus, we conclude that learning an auxiliary network for reward shaping, or at least our current implementation thereof, is not suitable for these sparse reward environments.

The only model that properly balanced exploration and exploitation was the potential-based learning method that repeatedly stepped back its start state from the end. This model tied Chat GPT by progressing past the first major obstacle of the game (finding a trap door underneath a rug) likely because later stages of the game feature more constrained traversal to teach the model how to seek rewards through both exploration and exploitation. We suspect that the combination of soft-actor updates and structured training data introduction with clear reward signals contributed to the success of this model.

Table 2: Qualitative Comparison for Zork

Reward Shaper	Buffer Style	Unique Actions	Mean Action Length	Unique States
Potential Based	Random	7	1.976	10
Potential Based	Curriculum (beginning)	8	2.0	29
Potential Based	Curriculum (end)*	6	1.001	19
Learned	Random	6	1.201	31
Learned	Curriculum (beginning)	**9**	1.407	**49**
Learned	Curriculum (end)*	5	1.191	38

*** critic pretrained for ten steps to ensure quality critic before actor steps**

Table 3: Most Common Actions Comparison for Zork

Reward Shaper	Buffer Style	Most Common Actions (number of instances)
Potential Based	Random	"east" (215), "take leaves" (210), "pull leaves" (197)
Potential Based	Curriculum (beginning)	"close mailbox" (219), "east" (211), "go around forest" (198)
Potential Based	Curriculum (end)*	"south" (216), "west" (215), "north" (200)
Learned	Random	"east" (206), "open mailbox" (201), "south" (200)
Learned	Curriculum (beginning)	"take leaves" (209), "south" (200), "northwest" (200)
Learned	Curriculum (end)*	"east" (214), "south" (208), "north" (196)

*** critic pretrained for ten steps to ensure quality critic before actor steps**

5.2 Qualitative Analysis

During the creation of the Chat GPT Zork baseline, it was clear that the LLM did not have sufficient memory to play the game competently. For example, it would often reference objects that it had seen but had not picked up. Additionally, the GPT model struggled to format commands in a way that was recognizable by the parser. Many commands were too long (e.g., "look around carefully") and it often struggled to revise rejected commands (e.g., from "lift rug" to "lift corner of rug" to "carefully lift up corner of rug" before eventually using "move rug" successfully). Furthermore, it took several minutes to take only a few actions, which is yet another downside of these large models.

Interestingly, our best scoring model (Potential-Based Reward Shaper with Curriculum Learning from near-end states) from Table 1 can be seen in Table 2 to take the second least number of unique actions and reach the second least number of unique states. From Table 3, we can clearly see that this model learns to preferentially favor strict directional actions ("south", "west", and "north") over other direct actions like grabbing items. In combination with its winning exploration score from Table 1, we can interpret this model as having best learned to simply explore the environment with minimal object manipulation. The Zork game has a highly sparse reward structure, making significant scoring from object manipulation unlikely due to the vast number of possible item combinations. In contrast, exploring the environment is more likely to yield *some* reward. End-based curriculum learning seems to emphasize this, as simply moving *somewhere* through the game often leads to more progress than trying to use objects without success. Note that the Learned Reward Shaper with the near-end Curriculum Learning also biases toward learning more directional action, supporting this hypothesis.

In general, we can see from Table 2 that our models learn relatively few actions and achieve relatively few states. Zork is a large game with many states and significantly many action combinations, so the small number of learned, best actions is suggestive of opportunity for improvement. While our winning model from Table 1 does score as high as Chat GPT, it and all our models still fail to approximate an ideal agent in this game, which would, like a professional human, use a significantly larger pool of actions to achieve significantly many more unique states.

6 Discussion

Given that our model using potential-based reward shaping experienced the most success, we take this opportunity to consider opportunities for models that achieve superior performance. Our first

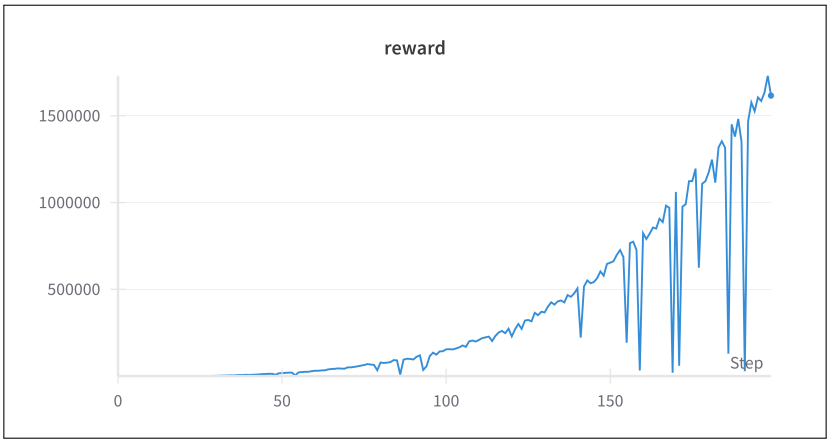

Figure 1: Visualization of the best model's reward.

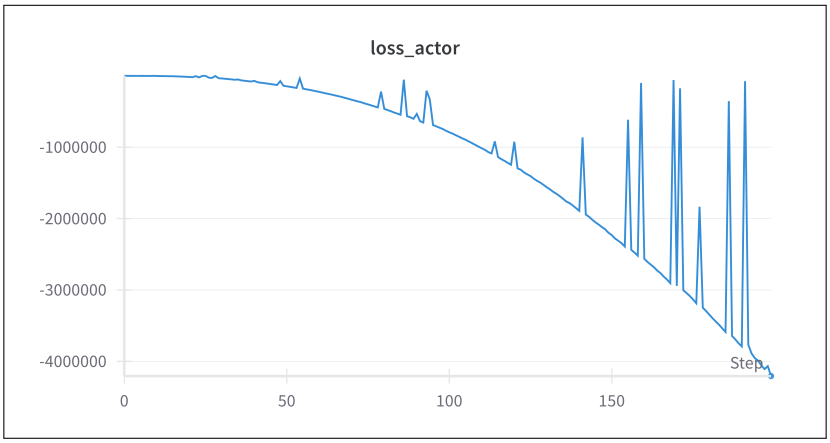

Figure 2: Visualization of the best model's actor loss.

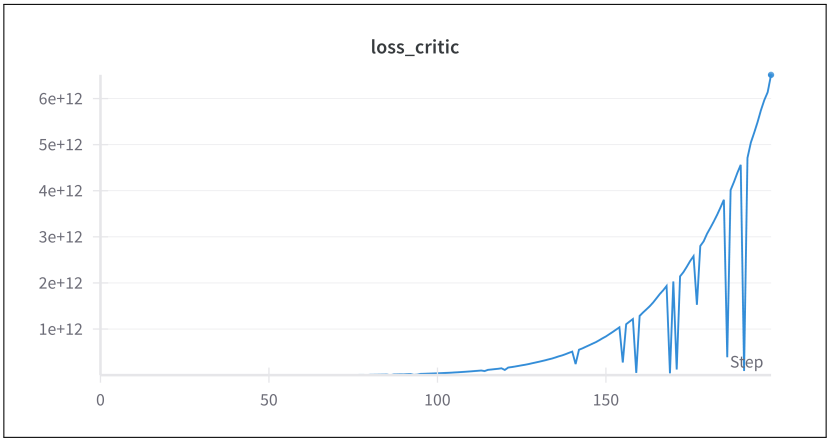

Figure 3: Visualization of the best model's critic loss.

consideration is several potential post-processing decisions that could improve our action selection. From our qualitative observations, we understand that our models tend to select repeated actions. Sometimes this is advantageous (e.g., you swing your sword at a troll, but you miss, so you try again) but this is often a mistake. As such, we could consider assigning a penalty to selecting repeated actions that decrements the similarity score and decreases the likelihood this action is selected. This had negligible impact on our current version of the model, given it is in the early stages of its training, but we expect this process could be beneficial if applied to a model that is further along in its training. We also experimented with normalizing our embedding vectors and using different measures of similarity with little success, so repetition is an open problem that we would hope to assess in future work.

Next, we would have wanted more time to fine-tune some of our hyperparameters, including model size, embedding size, and soft update weight. We suspect that our current embedding size may be too large for our model to sufficiently learn, but we wanted to keep this parameter constant to compare the effects of different reward shapers and replay buffer styles. As such, subsequent experiments could use smaller embeddings or techniques like PCA or t-SNE dimensionality reduction to see if this improves our model's ability to learn.

Another logical step would be to test this agent on other games and see if its knowledge transfers to other settings. Since these models are not trained for sufficient time on Zork, we expect they would also perform poorly on other games. However, an ideal test of a subsequent model would be to test it on a different game and see if its problem-solving skills generalize well. In some initial attempts, we found our current models to perform very poorly on other games, so we consider a multi-game training loops as a potential next step once we find a system that works reasonably for a single game.

7 Conclusion

We are pleased to report that we have successfully trained a SAC model to replicate the performance of state-of-the-art LLMs on the text-based adventure game Zork. We accomplished this by trying various reward shaping and curriculum learning methods, finding the use of endgame states and a potential-based reward shaper to have high potential. We believe this is because these choices give the model significant reward signals at the beginning of training to begin to adequately balance exploration and exploitation actions.

Future work may include more sophisticated language models and RL methods better suited for sparse reward, like Dreamer-v3 or even goal-conditioned RL. However, we believe that this project represents an encouraging step toward surpassing LLMs at general problem-solving without using large and expensive architectures. We hope that subsequent work considers the importance of curriculum learning and potential-based reward shaping for sparse reward text-based problem solving environments.

8 Team Contributions

- **Javo:** Led the qualitative evaluation of the trained models, including methods of extracting quality responses (e.g., nucleus sampling). Trained several intermediary models.
- **Phil:** Generated data from the Jericho dataset for offline training, wrote the soft actor critic implementation and initial versions of reward shaping methods. Helped debug and run online training loop and evaluation scripts.
- **Nate:** Performed hyperparameter analysis, debugging and running training loop. Trained several final models. Performed some qualitative analysis.

We believe that our team equitably balanced our work and has delivered a project that required three people to complete.

Changes from Proposal Our initial goal was to assess several types of text games, but it was much more challenging to develop a working model on human-made games than initially expected. As such, we focused on the human-made games and developing low-cost methods of playing these games without the use of expensive LLM architectures.

References

Prithviraj Ammanabrolu and Mark O. Riedl. 2019. Playing Text-Adventure Games with Graph-Based Deep Reinforcement Learning. https://doi.org/10.48550/arXiv.1812.01628 arXiv:1812.01628 [cs].

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. 41–48.

Lebling, Blank, and Anderson. 1979. Special Feature Zork: A Computerized Fantasy Simulation Game. *Computer* 12, 4 (1979), 51–59. https://doi.org/10.1109/MC.1979.1658697

Weichen Li, Rati Devidze, and Sophie Fellenz. 2023. Learning to Play Text-Based Adventure Games with Maximum Entropy Reinforcement Learning. In *Machine Learning and Knowledge Discovery in Databases: Research Track*, Danai Koutra, Claudia Plant, Manuel Gomez Rodriguez, Elena Baralis, and Francesco Bonchi (Eds.). Springer Nature Switzerland, Cham, 39–54. https://doi.org/10.1007/978-3-031-43421-1_3

Philip Osborne, Heido Nõmm, and André Freitas. 2022. A Survey of Text Games for Reinforcement Learning Informed by Natural Language. *Transactions of the Association for Computational Linguistics* 10 (Aug. 2022), 873–887. https://doi.org/10.1162/tacl_a_00495

Dongwon Kelvin Ryu, Meng Fang, Shirui Pan, Gholamreza Haffari, and Ehsan Shareghi. 2023. A Minimal Approach for Natural Language Action Space in Text-based Games. https://doi.org/10.48550/arXiv.2305.04082 arXiv:2305.04082 [cs].

Chen Feng Tsai, Xiaochen Zhou, Sierra S. Liu, Jing Li, Mo Yu, and Hongyuan Mei. 2025. Can Large Language Models Play Text Games Well? Current State-of-the-Art and Open Questions. https://doi.org/10.48550/arXiv.2304.02868 arXiv:2304.02868 [cs].

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. Keep CALM and Explore: Language Models for Action Generation in Text-based Games. https://doi.org/10.48550/arXiv.2010.02903 arXiv:2010.02903 [cs].

Mikuláš Zelinka, Xingdi Yuan, Marc-Alexandre Côté, Romain Laroche, and Adam Trischler. 2020. Building Dynamic Knowledge Graphs from Text-based Games. https://doi.org/10.48550/arXiv.1910.09532 arXiv:1910.09532 [cs].

A Implementation Details

Our GitHub repository with all of our code, model checkpoints, and evaluation data is available at https://github.com/pabaill/text-game-rl. All team members contributed to the repository, as several commits were anonymous from EC2 instances.