Solving TextWorld using Rollout based LLM

Aakash Sudhirbhai Vora
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ, USA
avora8@asu.edu

Pavel Dolin
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ, USA
pdolin@asu.edu

ABSTRACT

Text-based games have been used for a long time to evaluate the abilities of agents in terms of textual comprehension and reasoning. However, solving these games still poses a challenge to the AI community. Also, they involve planning and decision-making aspects which further complicates the problem. We propose to explore the use of online training algorithms like rollout in conjunction with Large Language Models to address this issue. We believe that textual capabilities combined with the cost improvement property of rollout should provide good results. We test our hypothesis on TextWorld games and see promising results using our approach.

KEYWORDS

Reinforcement Learning, Transformers, GPT, Rollout, Text-based Games, TextWorld

1 INTRODUCTION

Games have been an integral part of the field of Artificial Intelligence. Since the early days of AI, games have been used to design agents with specific skills. Games provide a way to approach a specific domain and tune in the difficulty level of the problem such that the skills of an intelligent agent can be easily and effectively. Games also act as a medium to compare the performance of such agents with humans. Unsurprisingly, the games have also been an important part of Reinforcement learning.

Text-based games are one of the important classes of games. Such games provide an interface to understand and interact with the environment through textual descriptions. They allow us to evaluate the textual comprehension, commonsense understanding, and logical capabilities of a player. Language is an inseparable part of human society. A majority of knowledge transfer happens through language and textual information. Thus, these text-based games have become a benchmark for evaluating the textual abilities of an algorithm.

Though text-based games have been around for a long time and many researchers have tried to develop algorithms that can play them, they are still found to be notoriously difficult. This is due to a set of skills required to master such games. The agent is expected to understand and comprehend the natural language which itself is a difficult task. Furthermore, the agent should also have a model of the real-world, commonsense, and the ability to use logic to reason. All these skills are hard to teach to an agent, so it expresses the popularity of text-based games among researchers.

However, due to the recent advent of the transformer-based model, the field of Natural Language Processing has seen a lot of new advances. These large language models(LLM) like BERT, T5, and GPT have shown surprising abilities in text-based problems like comprehension, generation, sentence classification, etc. Further, larger versions of GPT models are showing human-like performance in many NLP-related tasks. As the size of these models increases, they show more complex and near-intelligent behaviors.

All these advances, ask an interesting question: Are these models capable enough to create agents that can play text-based games and show improved performance? We plan to find an answer to this question in the present research project. We want to analyze whether such models have developed enough understanding of natural language to comprehend a description of some situation, make decisions and take actions.

2 RELATED WORK

The problem of solving text-based games have been addressed using a number of different approaches. Narasimhan et al. (2015)[6] introduced an approach called LSTM-DQN, where a Long Short-Term Memory (LSTM) is used to incorporate textual representations of states and actions into vector representations. The Q-values for state-object and state-action pairs were then estimated separately using these vectors as input to a Deep Q-Network (DQN).

He, Ji, et al.(2015)[4] proposed another approach called DRRN, which stands for Deep Recurrent Reinforcement Network. In this approach, a deep neural network is used to learn the embeddings of state and action texts. Based on a similarity metric, such as inner product, between the state and action vectors, the action is selected.

A more recent approach was presented by Kimura, Daiki, et al., (2021)[5], where they combined ConceptNet, a sizable commonsense knowledge graph, with first-order logic to produce predicate representations of the universe. These representations were then used as input to a Logical Neural Network (LNN), which used them to choose the appropriate action.

However, we have identified areas for improvement in these previous approaches. Firstly, instead of relying on LSTM-based or Bag-of-words representations for textual information, we suggest using attention-based Large Language Models (LLMs) like BERT to construct more robust and contextualized representations of textual information rather than LSTM-based or Bag-of-words representations. These models have demonstrated outstanding performance across a range of natural language processing tasks, and they may be able to offer superior representations for complicated task decision-making.

Another aspect that has been overlooked in previous approaches is the incorporation of planning into the decision-making process. In games or tasks with dynamic settings, planning is essential to success because it enables players to anticipate future conditions and make intelligent actions. However, most previous approaches lack proper consideration of planning. To address this limitation,

Figure 1: Left: Map of the typical game in TextWorld environment. Right: Example of an interaction with the game

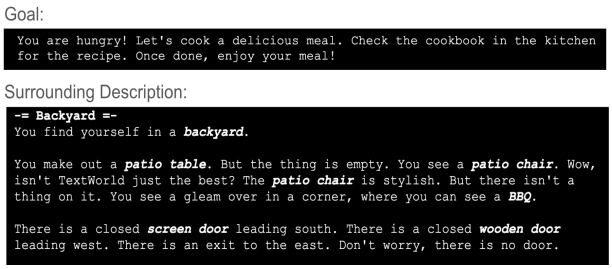

Figure 2: Goal and surrounding description from TextWorld

we propose the use of rollout algorithms together with LLM-based policies.

3 BACKGROUND

In this section, we would introduce the TextWorld[3] games environment and cover the necessary background on the Large Language model.

3.1 TextWorld

TextWorld is an open-source, text-based game framework that can be used for the generation and simulation of a game. TextWorld was created by the Microsoft research team. The main goal of TextWorld is to provide an environment that can help to easily train and test RL agents for text and planning-related capabilities.

As a typical setting in TextWorld games, the agent is present in a house and is supposed to perform a series of tasks to complete the quest. At the start of the game, the agent is provided with a description of the room of the house it is in and a detailed description of the surroundings like bed, furniture, doors, etc. Further, the initial description also included the list of tasks the agent needed to perform in order to complete the quest. The environment is designed in such a way that the agent can interact with its surroundings by executing a command such as "open the fridge" and the game would respond by explaining the effects of the action for example "You open the fridge" or "The fridge is already open". The environment provides the list of admissible commands that an agent can execute at each step. The agent can or can not be allowed to access the list of commands depending on the design of the algorithm.

TextWorld provides a number of text games, each with varying degrees of difficulty. It contains a class of games called Simple games which involve agents navigating through the house, performing a series of actions like picking up an item, opening a locked door using keys, opening a drawer or trunk, etc to complete the quest. Coin Collector class of games is similar to Simple games with the only difference being that the agent needs to collect a coin in the house while navigating through a number of potentially distracting rooms. Treasure hunter games have a similar concept to Coin collector games, except the agent needs to collect a specific item described in the initial description. Further, there is The Cooking Game type of game where the agent needs to navigate through the house to collect ingredients and use them to prepare a recipe. Finally, TextWorld also provides a way to generate custom games according to the requirement.

3.2 Large Language Model(LLM)

Large Language Models(LLM) in principle are neural networks with a large number of parameters that are capable of working with textual data. The architecture of such models includes a large number of transformer blocks stacked together. The group of transformer blocks can be interpreted as Encoder or Decoder blocks depending on the architecture. For example, BERT and T5 contain a series of transformer blocks as Encoder followed by another series of blocks forming the decoder. On the other hand, the GPT model contains only Decoder components.

These models have been very popular in natural language processing due to their surprising capabilities on NLP tasks. They exhibit a variety of skills like the ability to generate coherent and meaningful texts, an understanding of a wide range of languages, and the capability to perform a number of NLP tasks like language modeling, sentence classification, sentiment analysis, and question answering.

Recently, models like ChatGPT have further pushed the boundaries of such models by showing capabilities that require an understanding of the text and also reason about it to a certain degree. Whether or not such models have a complete understanding of reason and whether they are intelligent enough to take planned decisions is not yet clear and it is an active field of research.

4 METHODOLOGY

In this section, we would discuss the proposed idea and its architecture. As mentioned in previous sections, we plan to find an agent who can play text-based games effectively using the LLM like GPT. For our analysis, we fixed the text-based games environment to TextWorld.

4.1 Overview

We propose an online training rollout-based algorithm for RL agents that can play TextWorld games. Controls for the look-ahead step of the rollout can be either the list of admissible controls provided by the environment or they can be generated by using the LLM. For the rollout algorithm, LLM is used as the base policy which selects the control that needs to be performed during the look-ahead.

4.2 Problem Formulation

TextWorld games can be formulated as a POMDP problem. The text description provided at each step of the episode acts as the partial information state x_t. This information combined with previous controls $u_0..u_{t-1}$ and $x_0..x_{t-1}$ serves as a complete state I_t.

$$I_t = (x_t, x_{t-1}, u_{t-1}, ..., x_0, u_0) \tag{1}$$

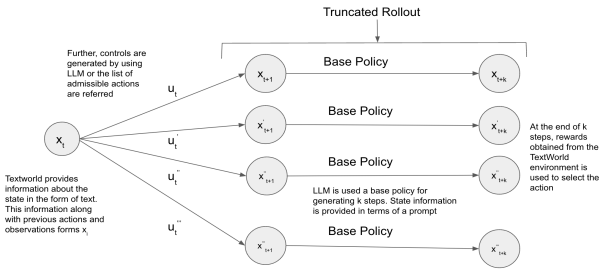

Figure 3: Proposed Architecture

At every step, the agent can select control u_t from the set of admissible control U_t and move to the next state x_{t+1}. Further, for every control, the environment provides a cost $c(x_t, u_t)$ to the agent with can 0,-1. The goal of the agent is to select controls during the episode that minimizes the cost for the entire trajectory. Bellman equation in term of I_t and U_t can be expressed as equation 2.

$$J(I_t) = \min_{u_t \in U_t} (c(x_t, u_t) + J(I_t + 1)) \tag{2}$$

4.3 Proposed Solution

As mentioned earlier, we plan to implement an online training rollout algorithm considering LLM as the base policy. We consider two separate setups of the games which differ in terms of whether the agent can access the list of admissible controls or not from the environment. However, the main architecture remains almost the same in both cases. When a TextWorld game is initialized, it provides information about the surroundings of the agent and a brief overview of the goals through a textual description. This information along with the list admissible list of controls (if accessible) is used to generate a set of controls that can be taken. These generated controls become the set of controls of one-step lookahead and rollout is executed for each of them till the horizon using LLM as the base policy. When using LLM to generate control at any stage (for lookahead or as the base policy), a prompt is used which contains all the information provided by the environment and also controls and observations from previous timestamps.

4.3.1 Control Generation for Lookahead. For the control generation for lookahead, if the agent is allowed access to the list of admissible controls at each timestamp then rollout is executed for each of the controls till the horizon and the control with the least trajectory cost is selected. However, if the agent is not allowed to access the admissible controls, the control space becomes very large. In such cases, LLM itself is used to generate k controls using a prompt and these controls are further used for the rollout step.

4.3.2 Prompt. LLM can be used for a certain task by prompting techniques. Basically, since LLM are generative models, they can only generate text. If we want to use them for a certain task like classification or prediction, then we need to generate a prompt which is a formulation of the problem such that an extension of the prompt must be the answer. For the current problem, we want to use LLM to generate probable controls that can lead to low costs. Furthermore, we also want to provide as much information about the environment as possible so that LLM can be conditioned on it to generate the best control. Hence, at each timestamp, we combined all the text output from the environment along with previous controls and observations to generate the prompt. If the agent has access to admissible control, then that information is also included in the prompt. The sample prompt used for the experiment is presented in the Table 1.

I am an expert game player. If my goal is 'Hey, thanks for coming over to the TextWorld today, there is something I need you to do for me. First of all, you could, like, look and see that the antique trunk inside the bedroom is opened. Then, recover the old key from the antique trunk. Then, make absolutely sure that the wooden door inside the bedroom is unlocked. After unlocking the wooden door, open the wooden door in the bedroom. Then, try to head east. After that, try to travel south. Once you get through with that, take the milk from the couch within the living room. Having taken the milk, attempt to travel north. That done, rest the milk on the stove inside the kitchen. And if you do that, you're the winner! -= Bedroom =- You make a grand eccentric entrance into a bedroom. You see a closed normal looking chest drawer right there by you. You can see an antique trunk. You see a king-size bed. But the thing is empty. There is a closed wooden door leading east. ' I took action 'inventory' previously and observed 'You are carrying nothing. ' I took action 'open chest drawer' previously and observed 'You open the chest drawer. ' My admissible actions are 'close chest drawer', 'examine antique trunk', 'examine chest drawer', 'examine king-size bed', 'examine wooden door', 'inventory', 'look', 'open antique trunk'. My best action (do not repeat previous actions) from list of admissible actions would be '

Table 1: A sample prompt for LLM

4.3.3 Cost/Rewards. The rewards obtained from TextWorld are independent of the timestamp. Basically, if the agent performs a certain control at the first timestamp or after the k timestamp, TextWorld still provides the same rewards. This leads to an issue since we want the agent to take the correct controls in a minimum number of timestamps. To enforce this constraint, we have adjusted the rewards according to the timestamps. For each control, the agent receives a reward of c_t/t instead of directly c_t.

4.3.4 Semantically similar controls. As discussed earlier, the control space becomes very large if the agent is not allowed to access the admissible controls. In such a situation, even if the agent selects the correct control, the environment might not understand it since it expects a paraphrased version of the control. To remediate such issues, we have added a semantic similarity model to the environment which can compare a given control with admissible controls and accept it if it is semantically matched with some admissible control with high confidence.

5 EXPERIMENT AND RESULTS

5.1 Experiment Setup

As mentioned in previous sections, the main goal of our research is to assess whether text-based games can be played by using large language models. In this section, we would discuss the experimental setup and the results obtained.

5.1.1 TextWorld Setup. We use TextWorld games as our testing environment. For all the experiments, we have fixed the following configurations. Firstly, we focused only on the Simple Games from all the available classes of games. For each execution, a new Simple Games was generated randomly which has a similar difficulty level but includes different objects and quest objectives. Secondly, the games are generated with detailed descriptions of the goals. Finally, the dense rewards are used i.e. rewards are given to an agent on accomplishing each task for the quest. We have executed two separate experiments each with and without access to the list of admissible control. If admissible controls are not available, 10 controls are generated by prompting LLM. The entire episode is executed for 10 timestamps. We execute truncated rollout with LLM base policy till the horizon of 5 timestamps with end reward at the truncation state as 0. For the semantic similarity model in the experiment without admissible controls, all-MiniLM-L6-v2 model[2] from hugging face library[1] is used. For measuring the similarity between two sentences, both of them are passed to all-MiniLM-L6-v2 model and encoded into vectors. The cosine similarity between these generated vectors is referred to as the similarity score. During the execution, for each admissible control, the similarity between it and the control provided by the agent is compared and ranked. If the highest-ranking admissible control has a similarity higher than a certain threshold, that admissible control is executed instead of the control provided by the agent. For our experiments, we have considered 0.7 as a threshold for similarity.

5.1.2 LLM Setup. As the large language model, we have used the GPT-J model. The GPT-J model is an open-source model trained by EleutherAI[7] and provided for use through a hugging face library. This model has 6 billion parameters with comparable accuracy to smaller models of GPT-3 trained by OpenAI. Further, due to computational restrictions, we have compressed the model to 8-bit precision using quantization.

5.2 Results

The table shows the average percentage of total reward collected by the rollout with LLM as the base agent in the 10 runs of an episode of 10 timestamps with or without access to the admissible controls. To create a baseline, the rewards obtained by using an agent that selects controls randomly and an agent with LLM as policy are also shown.

Agent	Avg. % of Total Reward (10 episodes)
Random Agent	14.9
GPT-J	10.18
Rollout with GPT-J	89

Table 2: Results for rollout with access to admissible controls

Agent	Avg. % of Total Reward (10 episodes)
Random Agent	-
GPT-J	9.11
Rollout with GPT-J	60.3

Table 3: Results for rollout without access to admissible controls

6 DISCUSSION

From the results, it can be observed using LLM that naive policy is not able to get good rewards. Given the results from the random agent, we can say that LLM's performance is almost equivalent to it. However, the high values of reward obtained by the rollout in both the setup highlight the drastic improvement obtained due to it. Furthermore, if the agent is not allowed to access the admissible controls, the rollout is still able to maintain a comparable performance highlighting its robustness.

However, the performance of the rollout comes at a cost. The rollout algorithm is always computationally expensive as compared to the base policy. Further, in the current context, we are using LLM as the base policy which itself is computationally heavy. Thus, the resultant rollout algorithm is very slow and highly computationally intensive. However, we can justify it by arguing that our approach is an online training algorithm and hence it does not require any kind of training in terms of value approximation or LLM as a policy which would be also computationally costly.

Also, we observed that the rollout agent faced an issue when it needs to select a control that would give rewards in the future. This is due to a truncated rollout with an end-state reward approximation. This issue was addressed in our experiment by using dense reward signals. Further, we also noticed that the rollout agent with access to admissible controls performs so well that it achieves similar results with even the random agent as a base policy. This phenomenon questions the use of LLM as a base policy itself. However, this is justified by the good result of the rollout agent without access to admissible controls since without using LLM it would be hard to select a few best controls from the unrestricted control space.

7 CONCLUSION AND FUTURE WORK

In conclusion, we would like to mention that in this research project, we present a novel technique of using rollout in conjunction with LLM to solve TextWorld games. The approach has shown promising results on the games as compared to the baselines. Furthermore, we test our approach on two experimental settings with different difficulties (access to admissible control) and establish that the approach is robust to changes in the environment.

In the future, we would like to explore how our approach performs on other classes of games offered by TextWorld and also on other text-based games. Also, we would like to compare our results with that of the state-of-the-art technique. We would like to also explore how the approach performs with other LLMs and study the relationship between the size of the model and their performance.

REFERENCES

[1] [n. d.]. Hugging Face. https://huggingface.co/. Accessed: 2023-04-25.

[2] [n. d.]. Sentence Transformer: all-MiniLM-L6-v2. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2. Accessed: 2023-04-25.

[3] Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. 2019. Textworld: A learning environment for text-based games. (2019). https://arxiv.org/pdf/1806.11532.pdf

[4] Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2015. Deep reinforcement learning with a natural language action space. *arXiv preprint arXiv:1511.04636* (2015). https://arxiv.org/pdf/1511.04636

[5] Daiki Kimura, Masaki Ono, Subhajit Chaudhury, Ryosuke Kohita, Akifumi Wachi, Don Joven Agravante, Michiaki Tatsubori, Asim Munawar, and Alexander Gray. 2015. Neuro-symbolic reinforcement learning with first-order logic. *arXiv preprint arXiv:2110.10963* (2015). https://arxiv.org/pdf/2110.10963

[6] Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941* (2015). https://arxiv.org/pdf/1506.08941

[7] Ben Wang. 2021. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. https://github.com/kingoflolz/mesh-transformer-jax.