From Scripts to RL Environments: Towards Imparting Commonsense Knowledge to RL Agents

Extended Abstract

Abhinav Joshi
IIT Kanpur
Kanpur, India
ajoshi@cse.iitk.ac.in

Areeb Ahmad
IIT Kanpur
Kanpur, India
areeb@iitk.ac.in

Umang Pandey
IIT Kanpur
Kanpur, India
umangp@iitk.ac.in

Ashutosh Modi
IIT Kanpur
Kanpur, India
ashutoshm@cse.iitk.ac.in

ABSTRACT

Text-based games provide a framework for developing natural language understanding and commonsense knowledge about the world in Reinforcement Learning (RL) based agents. Existing text-based environments often rely on fictional situations and characters to create a gaming framework and are far from real-world scenarios. In this paper, we introduce `ScriptWorld`: A text-based environment for teaching agents about real-world daily chores and hence imparting commonsense knowledge. To the best of our knowledge, it is the first interactive text-based gaming framework that consists of daily real-world human activities created using scripts dataset. We release the gaming environment and perform a detailed analysis of the proposed environment: https://github.com/Exploration-Lab/ScriptWorld.

KEYWORDS

Text Based Games; RL; NLP; Commonsense Knowledge

ACM Reference Format:
Abhinav Joshi, Areeb Ahmad, Umang Pandey, and Ashutosh Modi. 2023. From Scripts to RL Environments: Towards Imparting Commonsense Knowledge to RL Agents: Extended Abstract. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 3 pages.

1 INTRODUCTION

Text-based games in reinforcement learning have attracted research interests in recent years [1, 2]. These games have been developed to impart natural language understanding (NLU) and commonsense reasoning capabilities in Reinforcement Learning (RL) algorithms-based agents. A typical text-based game consists of a textual description of states of an environment where the agent/player observes and understands the game state and context using text and interacts with the environment using textual commands (actions). For successfully solving a text-based game, in addition to language understanding, an agent needs complex decision-making abilities, memory, planning, questioning, and commonsense knowledge [1].

Existing text-based gaming frameworks (e.g., Jericho [2], and Text-World [1]) provide a rich fictional setup (e.g., treasure hunt in a fantasy world) and require an agent to take complex decisions involving language and fantasy world knowledge. However, the existing text-based frameworks are created using a fixed prototype and are often distant from real-world scenarios. Though these frameworks aim to provide a rich training bench for enhancing natural language understanding in RL algorithms, the fictional concepts in these games are not well grounded in real-world scenarios, making the learned knowledge non-applicable to the real world. In contrast, for trained RL algorithms to be of practical utility, they should be trained in real-world scenarios that involve daily human activities. Humans carry out daily activities without much effort by using implicit *Script knowledge* [4–6, 8]. Formally, **Scripts** are defined as sequences of actions describing stereotypical human activities [8]. Scripts entail implicit knowledge about the world. For example, when someone talks about "Washing Dishes", there lies an implicit knowledge of fine-grained steps which would be present in the activity. By just saying, "I washed dishes on Thursday," a person conveys the implicit knowledge about the entire process, like 1) taking dirty dishes to the sink, 2) running water into the sink, 3) adding soap, 4) scrubbing the dishes 5) rinsing and 6) drying over the rack. The implicit detailed understanding of a task not only helps to learn about an activity but also facilitates taking suitable actions depending on the environment and past choices. Moreover, for learning a new task, humans can quickly and effortlessly discover new skills for performing the task either by their knowledge about the world or reading about it (reading a manual). With the aim to promote similar learning behavior in reinforcement learning algorithms, in this paper, we propose **`ScriptWorld`**, a new text-based game environment based on real-world scenarios involving script knowledge. In this work, we explore if script knowledge from existing script corpus can come in handy to develop a rich text-based environment that contains the human understanding of daily chores and are close to real-life scenarios.

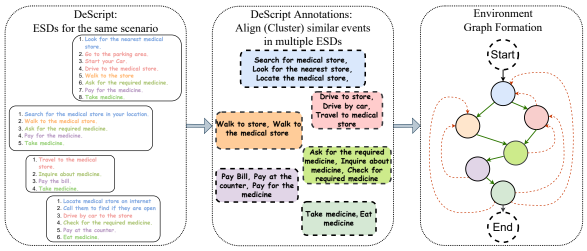
DeScript:
ESDs for the same scenario
1. Look for the nearest medical store.
2. Go to the parking area.
3. Start your Car.
4. Drive to the medical store.
5. Walk to the store
6. Ask for the required medicine.
7. Pay for the medicine.
8. Take medicine.
1. Search for the medical store in your location.
2. Walk to the medical store.
3. Ask for the required medicine.
4. Pay for the medicine.
5. Take medicine.
1. Travel to the medical store.
2. Inquire about medicine.
3. Pay the bill.
4. Take medicine.
1. Locate medical store on internet
2. Call them to find if they are open
3. Drive by car to the store
4. Check for the required medicine.
5. Pay at the counter.
6. Eat medicine.
DeScript Annotations:
Align (Cluster) similar events in multiple ESDs
Search for medical store, Look for the nearest store, Locate the medical store,
Drive to store, Drive by car, Travel to medical store
Walk to store, Walk to the medical store
Ask for the required medicine, Inquire about medicine, Check for required medicine
Pay Bill, Pay at the counter, Pay for the medicine
Take medicine, Eat medicine
Environment Graph Formation
Start
End

Figure 1: The figure shows a simplified version of the scenario, "get medicine," and the process of creating an environment graph (right diag.) from the ESDs (left diag.) and aligned events (middle diag.). The green directed edges represent the correct paths, and the red edges denote the transition when a wrong option is selected.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom. © 2023 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

2 SCRIPTWORLD ENVIRONMENT

ScriptWorld tries to bridge the gap between real-world scenarios (via Scripts) and text-based games for RL by creating a suitable environment. We take into consideration three design choices for developing the environment: **1) Complexity:** The game environment should be complex enough to test an RL algorithm's capacity to capture, understand and remember reasonable steps required for performing a daily chore. **2) Flexibility:** For an environment to help develop and debug RL algorithms, it becomes imperative to consider flexibility as a feature. The environment should be flexible in terms of difficulty levels and handicaps (hints) to provide a good test bench for reinforcement learning algorithms. **3) Relation to Real-World scenarios:** The environment should consist of activities/tasks that are grounded in the real world and are well understood among humans. The ScriptWorld environment is created from scratch using Python. A typical game begins by providing the agent with a quest (goal). The quest/goal is a one-line description of the scenario (e.g., plant a tree). The agent is also provided with initial observations (in English). Since it is a choice-based game, at each step in the game, the agent is also presented with a list of actions (in English) that it could opt to advance toward the goal. Based on the action selected by the agent, it is awarded a zero/positive/negative reward at each step. Every correct action takes the agent closer to task completion, whereas every wrong action results in a deviated path.

Graph Formation: We use an existing scripts corpus (*DeScript* [10]) for creating ScriptWorld environment. DeScript provides set of aligned Event Sequence Descriptions (ESDs) ($\mathcal{E}_1^{\mathcal{S}_i}, \mathcal{E}_2^{\mathcal{S}_i}, \ldots, \mathcal{E}_N^{\mathcal{S}_i}$) for a scenario $\mathcal{S}_i$. Each ESD $\mathcal{E}_k^i$ consists of sequence of short event descriptions describing the activity: $\mathbf{e}_1^{(\mathcal{E}_k^i)}, \mathbf{e}_2^{(\mathcal{E}_k^i)}, \ldots \mathbf{e}_n^{(\mathcal{E}_k^i)}$. Gold alignment (done by humans) in DeScript results in events in different ESDs that are semantically similar, getting linked to each other, i.e., clustered together. For example, for the *Washing Dishes* scenario, events "put dishes in sink" ($\mathbf{e}_1^{(\mathcal{E}_1^{Wash})}$) in $\mathcal{E}_1^{Wash}$ and "take dirty dishes to sink" ($\mathbf{e}_1^{(\mathcal{E}_2^{Wash})}$) in $\mathcal{E}_2^{Wash}$ are linked (clustered) together. Aligned events (from different ESDs) are used to create a graph having nodes as the event clusters (of aligned events) and directed edges representing the prototypical order of the events. In particular, a directed edge is drawn from node p to q if there is at least one event in node p that directly precedes an event in node q.

Environment Creation: We generate the game environment using the created graphs (Figure 1). For each state in the environment, the agent is required to pick the correct action (choice) from the available options. Since the created graph contains a wide variety of suitable actions grouped in a node, we sample the right choice from the available actions in a node. To create incorrect choices, we exploit the temporal nature of the graphs. As a graph contains a sequence of actions to perform a specific sub-task, all actions in nodes that are far from the current node become invalid for the current state. For selecting this node distance, we manually experiment with different node distances and find the different distances ($d_1, d_2, \ldots d_{10}$) suitable for sampling the invalid actions, i.e., for a scenario i, we consider all nodes at a distance greater than d_i hops from the current node. This strategy of sampling the invalid choices makes the environment more complex as all the options are related to the same scenario, and an understanding of event order in a task is required to achieve the goal.

Rewards: For all the scenarios, every incorrect action choice results in a negative reward of -1, and every correct choice returns a 0 reward. For task completion, the agent gets a reward of 10. , i.e., a player gets a maximum reward of 10 at the end of each game if they choose all the correct sequences of actions. The game terminates when an agent chooses 5 (env. setting parameter) successive wrong actions with a negative reward of -5.

Flexibility: To introduce flexibility in ScriptWorld, we consider two settings in a game. **1) Number of choices:** At each step, the number of choices presented to an agent can be changed (1 correct choice and rest all incorrect). As the number of options increases, it becomes more challenging for an agent to choose the right action. **2) Number of backward hops for wrong actions:** We choose the number of backward hops as another game setting that decides how many hops to displace whenever a wrong action is selected. When an agent selects an incorrect choice, its location is displaced by hopping it backward in the temporal domain, this back-hop distance is another parameter in the environment. In our experiments, agents played with the environment with a back-hop distance of 1. If an agent is teleported to a previous node (in the case of a wrong action), due to the presence of parallel paths in the graph, it may not follow the same route again. These two parameters introduce flexibility in our environment, giving the freedom of creating a suitable test bench for RL algorithms.

Handicaps (Hints): Text-based games are often complex for RL agents playing it from scratch. To mitigate the complexity issue in our environment, we introduce a version of the game with hints (referred as handicaps) for each state. The hint for a state provides a short textual clue for the next action to take at the current state. The presence of hints in the environment makes the gameplay relatively easier. Hints are generated automatically using GPT2 [7]. Scenario title concatenated with state node event description (both separated by a full-stop) is given as the prompt to GPT2 for generating a large number of hints and then a hint is sampled from them. We manually examined the hints to make sure that they do not repeat (verbatim) any of the existing actions. To introduce variability, one could also stochastically decide to show a hint, e.g., by sampling from a Bernoulli distribution at each state. However, in this paper, we consider only the setting where hints are shown at every state. More details about ScriptWorldcan be found at [3].

RL Agents: In the future, we plan to extensively experiment with a variety of RL agents (e.g., DQN, A2C, R-PPO, and PPO) in different settings. It would also be interesting to study the use of language models for imparting prior knowledge to RL agents[9].

3 CONCLUSION

In this paper, we propose a realistic environment (ScriptWorld) for teaching NLU capabilities and commonsense knowledge to RL agents. We plan to enrich our environment with more real-world scenarios using other existing script datasets. We also plan to make the environment parser-based, wherein an agent can take action by specifying in the form of free-form text, and we plan to use LMs for parsing the text.

REFERENCES

[1] Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. 2018. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*. Springer, CoRR, CoRR, 41–75.

[2] Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. AAAI, New York, USA, 7903–7910.

[3] Abhinav Joshi, Areeb Ahmad, Umang Pandey, and Ashutosh Modi. 2023. ScriptWorld. https://github.com/Exploration-Lab/ScriptWorld

[4] Ashutosh Modi. 2016. Event Embeddings for Semantic Script Modeling. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Berlin, Germany, 75–83. https://doi.org/10.18653/v1/K16-1008

[5] Ashutosh Modi and Ivan Titov. 2014. Inducing Neural Models of Script Knowledge. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Ann Arbor, Michigan, 49–57. https://doi.org/10.3115/v1/W14-1606

[6] Ashutosh Modi, Ivan Titov, Vera Demberg, Asad Sayeed, and Manfred Pinkal. 2017. Modeling Semantic Expectation: Using Script Knowledge for Referent Prediction. *Transactions of the Association for Computational Linguistics* 5 (2017), 31–44. https://doi.org/10.1162/tacl_a_00044

[7] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[8] Roger C Schank and Robert P Abelson. 1975. Scripts, plans, and knowledge. In *IJCAI*, Vol. 75. International Joint Conferences on Artificial Intelligence Organization, Georgia, USSR, 151–157.

[9] Ishika Singh, Gargi Singh, and Ashutosh Modi. 2022. Pre-trained Language Models as Prior Knowledge for Playing Text-based Games. In *AAMAS*. IFAAMAS, Auckland, New Zealand, 1729–1731.

[10] Lilian D. A. Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. 2016. A Crowdsourced Database of Event Sequence Descriptions for the Acquisition of High-quality Script Knowledge. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. European Language Resources Association (ELRA), Portorož, Slovenia, 3494–3501. https://aclanthology.org/L16-1556