Fire Burns, Sword Cuts: Commonsense Inductive Bias for Exploration in Text-based Games

Dongwon Kelvin Ryu[♠] **Ehsan Shareghi**[♠ ♣] **Meng Fang**[♡]
Yunqiu Xu[◇] **Shirui Pan**[♠] **Gholamreza Haffari**[♠]
[♠] Department of Data Science & AI, Monash University
[♡] Eindhoven University of Technology [◇] University of Technology Sydney
[♣] Language Technology Lab, University of Cambridge
firstname.lastname@monash.edu m.fang@tue.nl
yunqiu.xu@student.uts.edu.au

Abstract

Text-based games (TGs) are exciting testbeds for developing deep reinforcement learning techniques due to their partially observed environments and large action spaces. In these games, the agent learns to explore the environment via natural language interactions with the game simulator. A fundamental challenge in TGs is the efficient exploration of the large action space when the agent has not yet acquired enough knowledge about the environment. We propose COMMEXPL, an exploration technique that injects external commonsense knowledge, via a pretrained language model (LM), into the agent during training when the agent is the most uncertain about its next action. Our method exhibits improvement on the collected game scores during the training in four out of nine games from Jericho. Additionally, the produced trajectory of actions exhibit lower perplexity, when tested with a pretrained LM, indicating better closeness to human language. [1]

1 Introduction

Text-based games (TGs) are environments where agents learn to comprehend situations in language and produce decisions in language (Hausknecht et al., 2020; Côté et al., 2018; Narasimhan et al., 2015). Deep Reinforcement Learning lends itself as a natural paradigm to solve TGs due to its ability to learn from unsupervised game playing experience. However, existing RL agents are far away from solving TGs due to their combinatorially large action spaces that hinders efficient exploration (Yao et al., 2020; Ammanabrolu and Hausknecht, 2020).

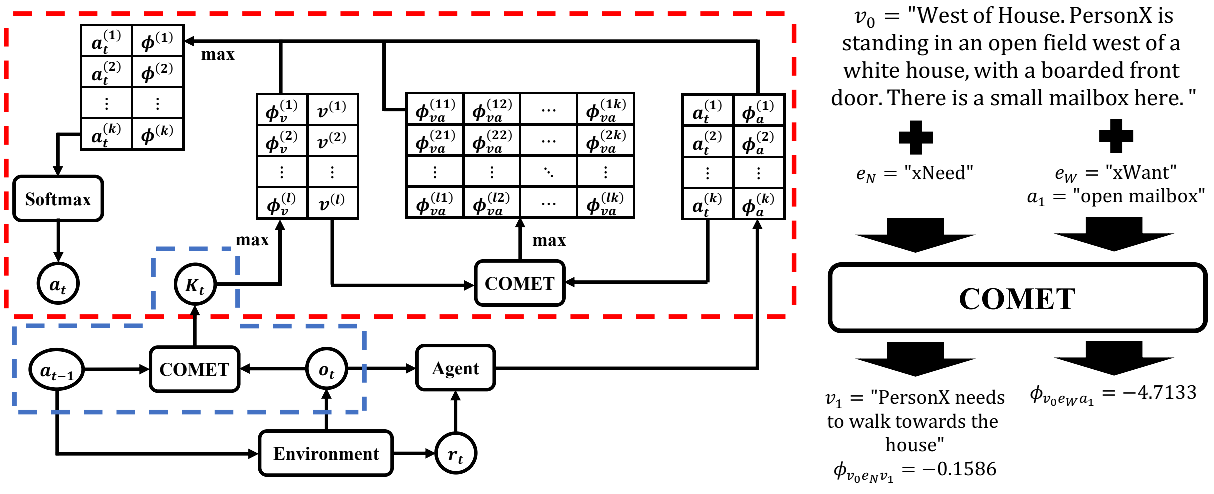
Ammanabrolu and Riedl (2019); Ammanabrolu and Hausknecht (2020) proposed incorporating a belief knowledge graph (BKG) built from the textual observations to help the agent reason more effectively about observed objects during the gameplay. Most of the recent works neglected linguistic aspects of TGs and focused on the construction and utilisation of BKG (Adhikari et al., 2020; Dambekodi et al., 2020; Xu et al., 2020; Ammanabrolu et al., 2020; Xu et al., 2021). Some exceptions involve developing pre-trained language models (LMs) to propose action candidates for a given observation (Yao et al., 2020), and investigating the relationship between semantic coherence and state representations (Yao et al., 2021).

In parallel, it has been argued that recent pretrained LMs capture commonsense factual knowledge about the world (Petroni et al., 2019; Kassner et al., 2021; Meng et al., 2021). More direct attempt in this direction was the commonsense transformer (COMET) which is a LM fine-tuned explicitly on commonsense knowledge graph (CSKG), to explicitly generate commonsense inferences (Bosselut et al., 2019; Hwang et al., 2021). Prior works with commonsense focused on completing BKG using pre-defined CSKG (Murugesan et al., 2020) or dynamic COMET-generated commonsense inferences (Dambekodi et al., 2020). Nonetheless, there is no work on explicitly using commonsense as an inductive bias in the context of exploration for TGs.

To bridge the gap, we propose *commonsense exploration* (COMMEXPL) which constructs a CSKG dynamically, using COMET, based on the state of textual observation per step. Then, the natural language actions are scored with COMET and agent, to re-rank the policy distributions. We refer to this as applying *commonsense conditioning*. However, doing this throughout the whole training is expensive and may not be beneficial as gameplay is not led by commonsense. To rectify this, we propose an *entropy scheduler*, driven by the entropy of the policy distribution, to regulate applying commonsense conditioning.

We demonstrate that our method encourages

[1] Code is available at https://github.com/ktr0921/comm-expl-kg-a2c



max
Softmax
max
max
COMET
COMET
Agent
Environment
v_0 = "West of House. PersonX is standing in an open field west of a white house, with a boarded front door. There is a small mailbox here."
e_N = "xNeed"
e_W = "xWant"
a_1 = "open mailbox"
COMET
v_1 = "PersonX needs to walk towards the house"
$\phi_{v_0 e_N v_1} = -0.1586$
$\phi_{v_0 e_W a_1} = -4.7133$

Figure 1: (Left) The overall architecture of COMMEXPL. The blue region is the *CSKG Construction* and the red region is *commonsense conditioning*. During *CSKG Construction*, COMET generates CKSG $\mathcal{K}$ given an action-observation pair while it produces node-to-action score given a node-action pair in *commonsense conditioning*. (Right) Example of how COMET works in COMMEXPL: Given a head node and edge, a tail node and its corresponding node-to-node score is generated while for node-to-action score, an action is passed as a desired tail node in COMET. Notations are defined in §2.

the agent to achieve higher game score during the training in four out of nine games in Jericho (Hausknecht et al., 2020). Furthermore, we show our method leads to producing more human-like natural language action. This is measured using the perplexity of the generated actions according to GPT-2 (Radford et al., 2019). We believe that natural language coherency/fluency is a crucial aspect of interactive intelligent agents (e.g. robots and dialogue systems) and hope our promising findings facilitate further developments of methods in this direction.

2 Approach

Notations. Text-based games are modelled as a partially observable Markov decision processes (POMDPs) of a tuple of $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{O}, \Omega, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$, $\mathcal{A}$, Ω denote sets of states, actions, and observations, respectively. Also, $\mathcal{R}$ and γ denote the reward function and the discount factor, while $\mathcal{P}$ and $\mathcal{O}$ denote the transition probabilities and set of conditional observations probabilities, respectively.

The agent requires to map an observation to a state ($\Omega \rightarrow \mathcal{S}$) and produce a policy π. By selecting an action a_t from the policy π, the agent changes current state s_t, receives a reward signal r, receives an observation through transition $\mathcal{P}(s_{t+1}|s_t, a_t)$, and also receives a conditional observation $\mathcal{O}(\Omega_t|s_t)$. The agent learns the policy $\pi_\theta(\boldsymbol{a}|\boldsymbol{o})$ that maximizes the expectation of the cumulative reward function $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$.

2.1 CSKG Construction

Let a CSKG be a graph $\mathcal{K} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes or vertices and $\mathcal{E}$ is a set of edges. The root node of CSKG requires to carry adequate information about the gameplay, so we amend the input to be the same format as how COMET is trained on, $\boldsymbol{v}_0 =$ "I " $+ \boldsymbol{a}_{t-1} +$ ". " $+ \boldsymbol{o}_t$ and replace all the "I" to "PersonX". To build CSKG we use COMET at every step of gameplay as a frozen commonsense generator to produce the tail node $\boldsymbol{v}_j$ given the head node $\boldsymbol{v}_i$ and edge $\boldsymbol{e}_j$ at time step t, formally denoted as $\Pr_\psi(v_{j,t}|\boldsymbol{v}_{j,<t}, \boldsymbol{v}_i, \boldsymbol{e}')$. Figure 1(Right) provides a visualisation of this. COMET takes v_0 as a head node and e_N as an edge and produces v_1 with the corresponding node-to-node score $\phi_{v_0 e_N v_1}$. Multiple tail nodes and node-to-node scores can be generated through the same input and based on the edge, the tail nodes vary dramatically. This process can be applied recursively to the tail nodes, expanding CSKG, i.e. generate tail nodes given v_1 head node with e_N. See Appendix A for more details.

2.2 Commonsense Conditioning

To blend commonsense into the agent's decision, the log-likelihood score is employed to contemplate each component independently. We, then, compute the total score as a weighted sum to promote the natural language action.

Agent-to-Action Score. The score function for the gameplay is obtained from the agent,

$$\phi_{\boldsymbol{a}}^{(k)} = \frac{1}{|\boldsymbol{a}_k|} \sum_{n=1}^{|\boldsymbol{a}_k|} \log \pi_\theta(a_{k,n}|a_{k,<n}, \boldsymbol{o}_{t-1}),$$

where $\phi_{\boldsymbol{a}}^{(k)}$ is the agent-to-action score for k action, computed as the sum of log-likelihood of the natural language action. Intuitively, the agent-to-action score signifies how much the action directs to the reward signals. This is learned during the online training of the agent.

Node-to-Action Score. Inspired by Bosselut et al. (2021); Yasunaga et al. (2021), the commonsense level of actions for each generated node is measured using COMET,

$$\phi_{\boldsymbol{v}_i\boldsymbol{e}_j\boldsymbol{a}_k} = \frac{1}{|\boldsymbol{a}_k|} \sum_{n=1}^{|\boldsymbol{a}_k|} \log \Pr_\psi(a_{k,n}|a_{k,<n}, \boldsymbol{v}_i, \boldsymbol{e}_j),$$

$$\phi_{\boldsymbol{va}}^{(lk)} = \max_{\boldsymbol{e}}(\phi_{\boldsymbol{v}_l\boldsymbol{e}_1\boldsymbol{a}_k}, \phi_{\boldsymbol{v}_l\boldsymbol{e}_2\boldsymbol{a}_k}, \cdots),$$

where $\phi_{\boldsymbol{v}_i\boldsymbol{e}_j\boldsymbol{a}_k}$ is the score per $\boldsymbol{va}$ edge, $\boldsymbol{e} \in \mathcal{E}_{\boldsymbol{va}}$, while the node-to-action score is denoted by $\phi_{\boldsymbol{va}}^{(lk)}$ which is the maximum $\phi_{\boldsymbol{v}_i\boldsymbol{e}_j\boldsymbol{a}_k}$ over $\boldsymbol{va}$ edges. The node-to-action score intersects commonsense with action, implying how plausible the action is given the commonsense prediction.

Node-to-Node Score. Additionally, we adopted the score between nodes in CSKG from Bosselut et al. (2021),

$$\phi_{\boldsymbol{v}_i\boldsymbol{e}'_j\boldsymbol{v}_l} = \frac{1}{|\boldsymbol{v}_l|} \sum_{n=1}^{|\boldsymbol{v}_l|} \log \Pr_\psi(v_{l,n}|v_{l,<n}, \boldsymbol{v}_i, \boldsymbol{e}'_j),$$

$$\phi_{\boldsymbol{v}}^{(l)} = \max_{\boldsymbol{v},\boldsymbol{e}'}(\phi_{\boldsymbol{v}_1\boldsymbol{e}'_1\boldsymbol{v}_l}, \phi_{\boldsymbol{v}_1\boldsymbol{e}'_2\boldsymbol{v}_l}, \cdots, \phi_{\boldsymbol{v}_2\boldsymbol{e}'_1\boldsymbol{v}_l}, \cdots),$$

where $\phi_{\boldsymbol{v}_i\boldsymbol{e}'_j\boldsymbol{v}_l}$ is the score per head node and $\boldsymbol{vv}$ edges, $\boldsymbol{e}' \in \mathcal{E}_{\boldsymbol{vv}}$, while the node-to-node score is $\phi_{\boldsymbol{v}}^{(l)}$, max of $\phi_{\boldsymbol{v}_i\boldsymbol{e}'_j\boldsymbol{v}_l}$ over head nodes and $\boldsymbol{vv}$ edges.[2] The node-to-node score is designed to promote commonsense triples that are more sensible commonsense-wise.[3]

Total Score. The total score assigned for each action is computed as:

$$\phi = \max_{\boldsymbol{v}}(\gamma_{\boldsymbol{a}}\phi_{\boldsymbol{a}} + \gamma_{\boldsymbol{va}}\phi_{\boldsymbol{va}} + \gamma_{\boldsymbol{v}}\phi_{\boldsymbol{v}}), \quad (1)$$

where ϕ is the total score per action since max is over nodes. The γ coefficients are hyperparameters and balance the weights between different components of the scoring function. Finally, the new conditioned policy is obtained as `softmax`(ϕ). We refer to this whole process as commonsense conditioning. A visualisation of the overall model is provided in the Figure 1(Left).

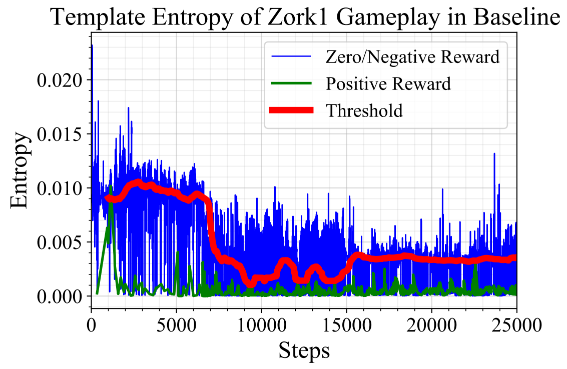


Figure 2: The plot of the entropy of `TEMPLATE` policy distribution over steps. The green indicates the entropy for a positive reward signal, and the blue does the same for zero or negative rewards. The entropy scheduler threshold of median is plotted as a red curve.

Intuitively, when the agent is not confident in current time-step, the policy distribution is arbitrary, resulting in homogeneous $\phi_{\boldsymbol{a}}$. This would be specifically the case during the initial stage of the training, but can also occur at any stage of the game where the agent cannot predict reward signal in a small number of steps. Under these circumstances, ϕ would be more dictated by $\phi_{\boldsymbol{va}}$ and $\phi_{\boldsymbol{v}}$. Conversely, when the agent is confident, the $\phi_{\boldsymbol{a}}$ for different actions will diverge and ϕ will be directed by both commonsense and the agent.

2.3 Entropy Scheduler

Since our technique uses a large LM for natural language generation, the main drawback with our approach is computational costs. In addition to this, where the agent is confident about acquiring the game score for a given action, commonsense could act as an undesired noise. To reflect on these, we propose the *entropy scheduler* to apply commonsense conditioning based on the confidence, the relative entropy of policy distribution. We collect the last 1000 number of the entropy of the template policy and apply commonsense conditioning if the current entropy is higher than the median. Figure 2 visualizes how the entropy scheduler works during training. This suggests that our entropy scheduler with a median threshold can apply commonsense conditioning to those actions with zero or negative

[2]A set of $\boldsymbol{va}$ edge and $\boldsymbol{v}$ edges can be different, but both are subset of CSKG edge set $\mathcal{E}_{\boldsymbol{vv}}, \mathcal{E}_{\boldsymbol{va}} \subseteq \mathcal{E}$.

[3]The example of adequate and poor commonsense phrases are: Given `PersonX lost umbrella`, `PersonX is angry` and `PersonX is hungry`, respectively.

Game	KG-A2C Score	KG-A2C PPL	KG-A2C + CommExpl Score	KG-A2C + CommExpl PPL	% Difference Score	% Difference PPL
balances	9.9	4.96	9.8	3.9	-1.01	**-21.37**
enchanter	19.6	4.47	19.6	3.73	0.0	**-16.56**
library	12.4	5.27	11.5	4.8	-7.26	**-8.92**
ludicorp	16.6	3.81	16.4	3.33	-1.2	**-12.6**
reverb	4.8	4.46	4.5	3.67	-6.25	**-17.71**
spirit	1.8	4.3	2.1	4.18	**16.67**	**-2.79**
zork1	24.7	3.77	30.7	3.49	**24.29**	**-7.43**
zork3	0.069	5.18	0.083	4.13	**20.29**	**-20.27**
ztuu	5.0	5.35	6.9	4.39	**38.0**	**-17.94**
MEAN					**+9.28**	**-13.95**

Table 1: Score and perplexity comparison over 9 game environments, with positive results highlighted by **boldface**. The score is computed as the average over the entire training to signify its performance during the training while perplexity (PPL) is measured for a given root node. The last column denotes the percentage difference between KG-A2C with and without CommExpl.

zork1: Kitchen. You are in the kitchen of the white house. A table seems to have been used recently for the preparation of food. A passage leads to the west and a dark staircase can be seen leading upward. A dark chimney leads down and to the east is a small window which is open.			
π	put down glass	open brown	put glass on table
$\hat{\pi}$	put glass on table	put down glass	go up
zork3: It is pitch black. You are likely to be eaten by a grue.			
π	put down lamp	take lamp	turn on lamp
$\hat{\pi}$	turn on lamp	put down lamp	go down

Table 2: An illustrative example of how action selection changes with CommExpl. Only top 3 actions are shown for readability. TEMPLATE policy is used for π, i.e. the TEMPLATE probability of put down OBJ is used for put down glass, while $\hat{\pi}$ is the policy conditioned on commonsense.

reward signals. [4]

3 Experiments

We use KG-A2C as our goal-driven baseline agent and compare it with KG-A2C with commonsense in a game suite of Jericho. A set of nine games are selected from Jericho carefully based on genre, including three daily puzzle games (library, ludicorp, reverb) and the rest six fantasy adventure games (balances, enchanter, spirit, zork1, zork3, ztuu). Both game setting and optimal configuration for KG-A2C in Ammanabrolu and Hausknecht (2020) were used in our experiments. *We reduced training steps to* $25,000$ *since our objective is to compare the quality of exploration during the training.* Only hyper-parameters in CommExpl have been optimized for fair comparison while all the parameters in COMeT were fixed during the training, resulting in the equal trainable parameters regardless of CommExpl. Details of the hyper-parameters and the experimental setup can be found in Appendix B.

3.1 Main Results

Similar to Ammanabrolu and Hausknecht (2020), we employed the optimal hyper-parameters fine-tuned on zork1 for nine games in Jericho. Table 1 shows the mean score across the entire training and the perplexity of the action given a root node. The score is to compare whether the agent with commonsense achieves *higher game score during the training*. Doing so implies how fast the agent learns with fewer steps, and therefore, more efficient exploration. Perplexity from LM is used as a metric for the smoothness of natural language action. We used GPT-2 from Huggingface (Wolf et al., 2020).

Score Table 1 shows that with CommExpl, the agent tends to acquire the game score more frequent in four gaming environments (spirit, zork1, zork3, ztuu). All four have at least 15% increases in game score during training. However, three environments (balances, enchanter, ludicorp) appear to gain no benefits from using CommExpl. On the other hand, the remaining two games (library, reverb) take commonsense negatively, suggesting that the commonsense from COMeT acts as a noise with respect to pursuing rewards. Per genre, interestingly, those daily puzzle games are either not influenced or negatively influenced from commonsense inductive bias while four out of six fantasy adventure games benefited from it. We speculate this might be due to the fine-tuning which was also done on a single game, zork1.

Coherency Table 1 shows that commonsense prior reduces perplexity of the natural language actions in all nine games. This is because, unlike the game score that is not directly related to commonsense, the semantic properties of the actions are directly related to commonsense. For environments like balances and reverb, despite the agent taking no benefits from commonsense,

[4] As shown in Appendix C, the training time still remains relatively long due to the natural language generation with a large COMeT.

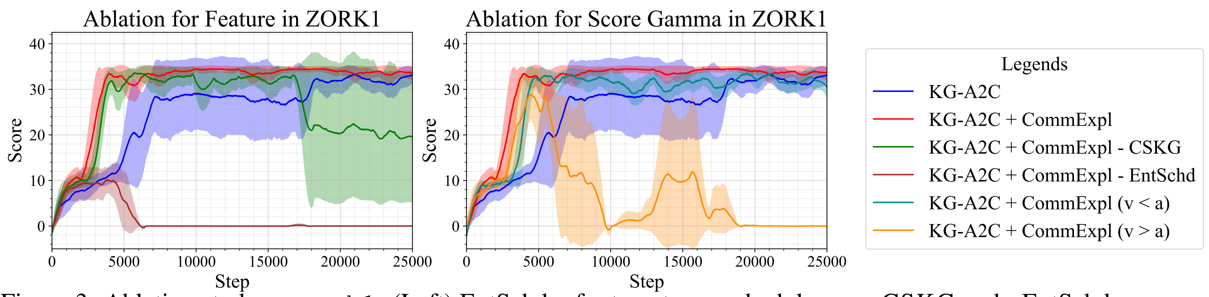


Figure 3: Ablation study on `zork1`. (Left) EntSchd refer to entropy scheduler, so - CSKG and - EntSchd mean we removed CSKG and entropy scheduler from COMMEXPL. (Right) '$>$' sign signifies how much commonsense (v) or agent (a) is weighted more over the other.

perplexity drops significantly (*e.g.*, $\sim$15%). This large reduction in perplexity also appears for fantasy games, in which `zork3` had $\sim$20% down and `spirit` took as little as $\sim$3% reduction. This suggests that the game takes advantages on the semantic coherency regardless of whether it helps to achieve high score of the game or the genre of the game.

Qualitative Samples Table 2 provides qualitative samples to show how natural language actions are re-ordered after commonsense conditioning. For instance, in the first example of `zork1`, COMMEXPL suppresses `open brown` and pushes `put glass on table` to the highest probability. In `zork3`, COMMEXPL promotes `turn on lamp` over others since the observation informs user that the surrounding is dark.

3.2 Ablation Results

We performed two ablation studies on `zork1` to obtain the optimal hyper-parameters. The first ablation study is for the absence of features, in which we removed CSKG construction and entropy scheduler completely. Thereafter, the changes in score gamma factors have been investigated. The γ coefficients are changed from $(\gamma_{\boldsymbol{v}} = 1, \gamma_{\boldsymbol{va}} = 0.7, \gamma_{\boldsymbol{a}} = 0.8)$ to $(0.4, 0.2, 1)$ for $(\boldsymbol{v} < \boldsymbol{a})$ model and $(1, 1, 0.3)$ for $(\boldsymbol{v} > \boldsymbol{a})$ model.

Feature Figure 3 (Left) shows that the absence of CSKG construction or entropy scheduler causes catastrophic forgetting. KG-A2C is prone to this regardless of commonsense because it does not use any memory component. However, injecting commonsense stochastically enhances the likelihood since the agent follows commonsense when it should not, i.e. a particular action is required to obtain game score. This overlaps with our motivation of entropy scheduler, that *the game score is not directly related to commonsense, so appropriate skipping is necessary*.

Dynamic CSKG contributes to a variety of commonsense, amplifying its commonsense reasoning, and a lack of this will provoke the agent acting more narrow with limited commonsense. Our plot shows that removing CSKG also contributes to the cause of catastrophic forgetting. This suggests that lack of diversity in commonsense may act as a noise to the exploration, and may push the agent to produce more skewed trajectories that cause failure. Therefore, the absence of any component leads to performance decay. Therefore, both are vital components in COMMEXPL.

Score Gamma Factor The contribution of the commonsense and the agent score is investigated on Figure 3 (Right). By increasing agent's gamma factor, the model acts more alike to the baseline than the optimal hyper-parameters since it trusts its own policy more. Conversely, adding more weights on commonsense leads to catastrophic forgetting. This is caused by the fact that the agent puts too much trust on commonsense, diverging from its own policy excessively. From these, we can conclude that the appropriate balancing is required to make exploration efficient and feasible.

4 Conclusion

We investigated the effect of commonsense in text-based RL agent during the training. Our results show that despite the hyper-parameters tuning on a single game, the proposed approach improves on other gaming environments in Jericho, total four out of nine. Furthermore, injecting commonsense also positively influences the semantics of natural language actions, resulting in lower perplexity. Our future work will extend its application to different text-based environments and investigate how this linguistic properties from LM helps the agent.

References

Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláŝ Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and William L. Hamilton. 2020. Learning dynamic belief graphs to generalize on text-based games. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Prithviraj Ammanabrolu and Matthew J. Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Prithviraj Ammanabrolu and Mark Riedl. 2019. Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 3557–3565. Association for Computational Linguistics.

Prithviraj Ammanabrolu, Ethan Tien, Zhaochen Luo, and Mark O. Riedl. 2020. How to avoid being eaten by a grue: Exploration strategies for text-adventure agents. *CoRR*, abs/2002.08795.

Antoine Bosselut, Ronan Le Bras, and Yejin Choi. 2021. Dynamic neuro-symbolic knowledge graph construction for zero-shot commonsense question answering. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 4923–4931. AAAI Press.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4762–4779. Association for Computational Linguistics.

Marc-Alexandre Côté, Ákos Kádár, Xingdi (Eric) Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. In *Computer Games Workshop at ICML/IJCAI 2018*, pages 1–29.

Sahith N. Dambekodi, Spencer Frazier, Prithviraj Ammanabrolu, and Mark O. Riedl. 2020. Playing text-based games with common sense. *CoRR*, abs/2012.02757.

Matthew J. Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7903–7910. AAAI Press.

Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. (comet-) atomic 2020: On symbolic and neural commonsense knowledge graphs. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 6384–6392. AAAI Press.

Nora Kassner, Philipp Dufter, and Hinrich Schütze. 2021. Multilingual lama: Investigating knowledge in multilingual pretrained language models. In *ACL*, pages 3250–3258.

Zaiqiao Meng, Fangyu Liu, Ehsan Shareghi, Yixuan Su, Charlotte Collins, and Nigel Collier. 2021. Rewire-then-probe: A contrastive recipe for probing biomedical knowledge of pre-trained language models. *arXiv preprint arXiv:2110.08173*.

Farhad Moghimifar, Lizhen Qu, Yue Zhuo, Mahsa Baktashmotlagh, and Gholamreza Haffari. 2020. CosMo: Conditional Seq2Seq-based mixture model for zero-shot commonsense question answering. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5347–5359, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Keerthiram Murugesan, Mattia Atzeni, Pushkar Shukla, Mrinmaya Sachan, Pavan Kapanipathi, and Kartik Talamadupula. 2020. Enhancing text-based reinforcement learning agents with commonsense knowledge. *CoRR*, abs/2005.00811.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Lisbon, Portugal. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International*

Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pages 2463–2473. Association for Computational Linguistics.

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):9.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, and Chengqi Zhang. 2021. Generalization in text-based games via hierarchical reinforcement learning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1343–1353, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, Joey Tianyi Zhou, and Chengqi Zhang. 2020. Deep reinforcement learning with stacked hierarchical attention for text-based games. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Shunyu Yao, Karthik Narasimhan, and Matthew Hausknecht. 2021. Reading and acting while blindfolded: The need for semantics in text game agents. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3097–3102, Online. Association for Computational Linguistics.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. Keep CALM and explore: Language models for action generation in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8736–8754, Online. Association for Computational Linguistics.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

A CSKG Construction

There are three different strategies for building the root node from the textual observation and the natural language action. The most generic one is, given $\boldsymbol{a}_{t-1}$ = “move rug" and $\boldsymbol{o}_t$ = “With a great effort, the rug is moved to one side of the room, revealing the dusty cover of a closed trap door.”, the root node is $\boldsymbol{v}_0$ = “PersonX ” $+ \boldsymbol{a}_{t-1} +$ “. ” $+ \boldsymbol{o}_t$ =“PersonX move rug. With a great effort, the rug is moved to one side of the room, revealing the dusty cover of a closed trap door.”. The example of CSKG with v_0 is in Figure A.1.

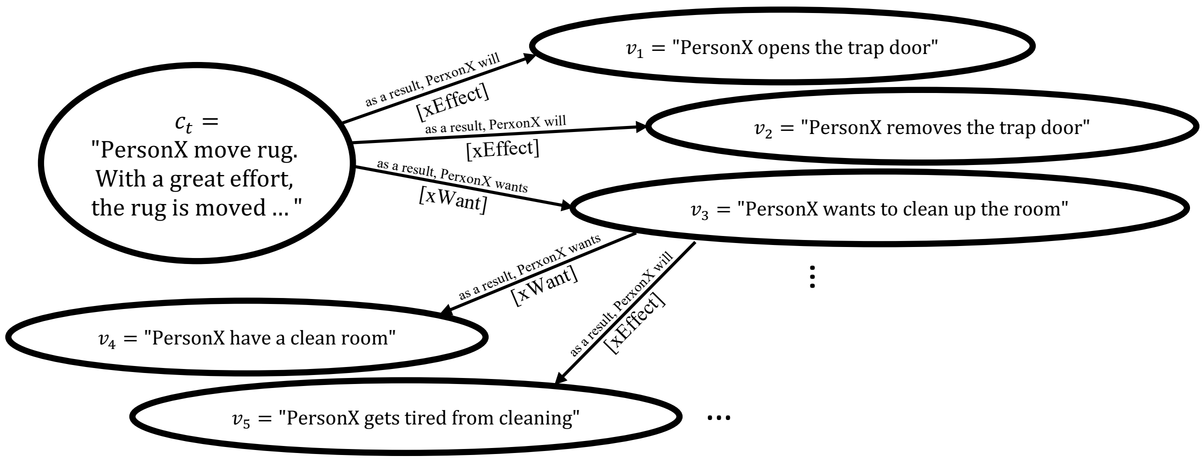
However, if the previous action a_{t-1} was not admissible, we set the room description of the textual observation as the root node. Finally, if the action is admissible, but the observation is too short (less than 20 tokens), the root node includes the previous room description of the textual observation at the beginning of the page, $\boldsymbol{v}_0 = \boldsymbol{o}_{\text{room},t-1} +$ “ PersonX ” $+ \boldsymbol{a}_{t-1} +$ “. ” $+ \boldsymbol{o}_t$.

These are motivated from 1) if the previous action is not admissible, the environment is not affected by it, so we simply use the previous room description that captures a lot of information about what the agent can do, 2) if the observation is too short that it does not carry enough information about the situation, we concatenate the previous room description to subjoin the information about surroundings, and 3) otherwise, the generic strategy to build the root node, the previous action and the consequence of it as textual observation.

B Experiment Setup

Action Sampling We set $n_{\texttt{TEMPLATE}}$ to be dynamic, only selecting those based on the probability threshold and validity. The threshold is calculated as 0.75 of its uniform distribution. For instance, zork1 contains 237 number of `TEMPLATE`, so the threshold is $0.75 \times \frac{1}{237} = 0.00316$. We only select the maximum of 7 `TEMPLATE` that exceeds the threshold. This avoids a large shift in policy distribution while attaining better computational efficiency. Additionally, we include valid templates to enforce the agent to act more towards on changing the world tree. We sampled objects like KG-A2C since KG-A2C already restricts objects and the actions are usually determined by the template. Therefore, $|\phi_{\boldsymbol{a}}| = n_{\texttt{TEMPLATE}}$, reducing the computations but still covering useful action sets.

Commonsense Transformer Our COMET is



c_t = "PersonX move rug. With a great effort, the rug is moved ... "
v_1 = "PersonX opens the trap door"
v_2 = "PersonX removes the trap door"
v_3 = "PersonX wants to clean up the room"
v_4 = "PersonX have a clean room"
v_5 = "PersonX gets tired from cleaning"
as a result, PerxonX will [xEffect]
as a result, PerxonX will [xEffect]
as a result, PerxonX wants [xWant]
as a result, PerxonX wants [xWant]
as a result, PerxonX will [xEffect]

Figure A.1: The CSKG construction from the corresponding root node with $\boldsymbol{a}_{t-1}$ = "move rug" and $\boldsymbol{o}_t$ = "With a great effort, the rug is moved to one side of the room, revealing the dusty cover of a closed trap door.". Each commonsense phrase node is presented as circle and a directed edge between them is CSKG edge.

BART fine-tuned on ATOMIC-2020 dataset, which is crowdsourced with natural language sentence nodes and 23 commonsense edges (Hwang et al., 2021). We assumed that the general COMET is still good enough to cover TGs. Since the gaming environment runs by the player character, we only focus on the social-interaction commonsense. "xNeed" and "xIntent" are chosen for CSKG construction, $\mathcal{E}_{\boldsymbol{vv}}$, since they deal with what is needed or intended for the event to occur, while "xWant" and "xEffect" for scoring the natural language actions, $\mathcal{E}_{\boldsymbol{va}}$, since they deal with what the player would do following the event. We further set $n_{\text{hop}} = 1$ and $n_{\text{gen}} = 2$ from the observation that they are good enough for zero-shot commonsense question answering (Bosselut et al., 2021; Moghimifar et al., 2020). During the online training of the agent, we freeze the parameters for COMET.

C Computational Expense

The number of node-to-node scores is directly related to the size of CSKG,

$$|\phi_{\boldsymbol{v}}| = \sum_{i=0}^{n_{\text{hop}}} (n_{\text{gen}} \times |\mathcal{E}_{\boldsymbol{vv}}|)^i,$$

where n_{hop} is the number of hops, n_{gen} is the number of triple generation and $\mathcal{E}_{\boldsymbol{vv}}$ is the edge space for CSKG.

On the other hand, the number of node-to-action scores is equal to the number of the total score ϕ,

$$|\phi_{\boldsymbol{va}}| = |\phi| = |\phi_{\boldsymbol{v}}| \times |\mathcal{E}_{\boldsymbol{va}}| \times |\phi_{\boldsymbol{a}}|,$$

where $\mathcal{E}_{\boldsymbol{va}}$ is the edge space for node-to-action score.

We assume $|\phi_{\boldsymbol{a}}| \approx 7$ since we select maximum of 7 templates with highest probability and valid templates. Therefore, in our setting, we can calculate the number of the natural language generations per step per environment as,

$$\begin{aligned} |\phi_{\boldsymbol{v}}| + |\phi_{\boldsymbol{va}}| &= |\phi_{\boldsymbol{v}}| + |\phi_{\boldsymbol{v}}| \times |\mathcal{E}_{\boldsymbol{va}}| \times |\phi_{\boldsymbol{a}}| \\ &= |\phi_{\boldsymbol{v}}| \cdot (1 + |\mathcal{E}_{\boldsymbol{va}}| \times |\phi_{\boldsymbol{a}}|) \\ &\approx \sum_{i=0}^{1} (2 \times 2)^i \cdot (1 + 2 \times 7) \\ &= 75 \end{aligned}$$

Finally, we can estimate the average number of natural language generation per step by multiplying the number of environments per step $n_{\text{env}} = 32$ and fraction from entropy scheduler $p \approx 0.5$,

$$\begin{aligned} (|\phi_{\boldsymbol{v}}| + |\phi_{\boldsymbol{va}}|) \times n_{\text{env}} \times p &\approx 75 \times 32 \times 0.5 \\ &= 1200 \end{aligned}$$

Throughout the training, we require to perform 1200 natural language generations using a large size COMET per step, so this increases the training time from $\times 3$ upto $\times 10$.