On the Effects of Fine-tuning Language Models for Text-Based Reinforcement Learning

Anonymous ACL submission

Abstract

Text-based reinforcement learning involves an agent interacting with a fictional environment using observed text and admissible actions in natural language to complete a task. Previous works have shown that agents can succeed in text-based interactive environments even in the complete absence of semantic understanding or other linguistic capabilities. The success of these agents in playing such games suggests that semantic understanding may not be important for the task. This raises an important question about the benefits of LMs in guiding the agents through the game states. In this work, we show that rich semantic understanding leads to efficient training of text-based RL agents. Moreover, we describe the occurrence of semantic degeneration as a consequence of inappropriate fine-tuning of language models in text-based reinforcement learning (TBRL). Specifically, we describe the shift in the semantic representation of words in the LM, as well as how it affects the performance of the agent in tasks that are semantically similar to the training games. These results may help develop better strategies to fine-tune agents in text-based RL scenarios.

1 Introduction

Text-based games (TBGs) are a form of interactive fiction where players use textual information to manipulate the environment. Since information in these games is shared as text, a successful player must hold a certain degree of natural language understanding (NLU). TBGs have surfaced as important testbeds for studying the linguistic potential of reinforcement learning agents along with partial observability and action generation. TBGs can be modeled as partially observable Markov decision processes (POMDP) defined by the tuple $\langle S, A, O, T, E, R\rangle$, where S is the set of states, A the set of actions, O the observation space, T the set of state transition probabilities, E is the conditional observation emission probabilities, and $R : S \times A \rightarrow \mathbb{R}$ the reward function. The agent's goal is to reach the end of the game by performing text actions, while maximizing the final score.

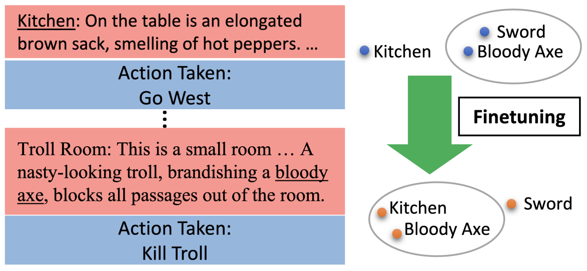

Kitchen: On the table is an elongated brown sack, smelling of hot peppers. ...
Action Taken: Go West
Troll Room: This is a small room … A nasty-looking troll, brandishing a bloody axe, blocks all passages out of the room.
Action Taken: Kill Troll
Kitchen
Sword
Bloody Axe
Finetuning
Kitchen
Bloody Axe
Sword

Figure 1: Semantic degeneration of the terms *kitchen* and *bloody axe* in *Zork 1*.

In TBGs, observations and actions are presented in the form of unstructured text, therefore, they must be encoded before being passed onto the RL network. Recent works in text-based RL adopt a strategy where such encoding is learned from the game, typically by fine-tuning a language model, such as embeddings or transformers, using the rewards values from the training (Yao et al., 2020; Wang et al., 2022a). We hypothesize that this approach may cause the language model to overfit the training games, leading to the degeneration of the semantic relationships learned during pretraining, and, subsequently, negatively impacting the agent's training efficiency and transfer learning capacity. We conduct experiments in two distinct TBG domains: (1) TextWorld Commonsense (TWC) (Murugesan et al., 2021a), and (2) Jericho (Hausknecht et al., 2019). The former provides a number of games where the goal is to perform house cleaning tasks such as taking objects from a location and placing them in their appropriate places, using commonsense knowledge. The latter provides a library of classic text-adventure games, such as the *Zork* (1977), each having its own unique objectives, characters, and events. Unlike TWC games, Jericho games may not let the player know a priori what the final goal is. Instead, the player is expected to explore the game to learn the story and complete the tasks one-by-one. In both domains, actions are selected from a list of possible moves.

Under this framework, we address the following research questions:

1. Does fine-tuning the language model to the RL rewards improve the training efficiency in comparison to fixed pre-trained LMs?

2. Does fine-tuning LMs make agents robust to tasks containing out-of-training vocabulary?

Our goal is to evaluate what are the implications, pros, and cons of fine-tuning LMs to the RL tasks. Our results indicate fine-tuning LMs to rewards leads to a decrease in the agent's performance and hinders its ability to play versions of the training games where the observations and actions are slightly reworded, such as through paraphrasing or lexical substitution (synonyms). In comparison to fixed pre-trained LMs, these fine-tuned agents under-performed in training and in test settings. We refer to this process as **semantic degeneration**, because it leads to loss of relevant semantic information, in the LM, that would be crucial to produce generalizable representation. For instance, by learning that the terms "bloody axe" and "kitchen" are related to each other in the game *Zork 1*, the agent overfits to this setting and, in turn, loses relevant information about "kitchen" and "bloody axe" that could be important to other games. In NLP, semantic generation might be an expected consequence of fine-tuning (Mosbach et al., 2020), however, the vast majority of text-based RL agents employ LMs that are fully fine-tuned to the game's semantics.

2 Background

Model and Architecture The general architecture of the agents in this work consist of a state encoder akin to the DRRN (He et al., 2015) with an actor-critic policy learning (Wang et al., 2016) and experience replay. The main components of the agent's network are (1) a text encoder, (2) a state-action encoder, and (3) an action scorer. The text encoder module is a language model that converts an observation $o \in O$ and action $a \in A$ from text form to fixed length vectors $f(o)$ and $f(a)$. The state-action encoder consists of a GRU (Dey and Salem, 2017) that takes as input the encoded state and actions, and predicts the Q-values for each pair: $Q_\phi(o, a) = g(f(o), f(a))$ given parameters ϕ. The action predictor is a linear layer that outputs the probabilities based on the Q-values from the previous layer. The chosen action is drawn following the computed probability distribution. The agent is trained by minimizing the temporal differences (TD) loss: $\mathcal{L}_{TD} = (r + \gamma \max_{a' \in A} Q_\phi(o', a') - Q_\phi(o, a))^2$ where o' and a' are the next observation and next actions sampled from a replay memory, γ is the reward discount factor.

Text Encoders We used three distinct types of encoders in this study:

- *Hash* - does not capture semantic information from the text. Follows the approach by Yao et al. (2021).
- *Word Embedding* - pre-trained static GloVe embeddings (Pennington et al., 2014) and a GRU to encode the sequences of tokens.
- *Transformers* - pre-trained LMs to encode observations (Devlin et al., 2018).

These encoders are often the top performer (Murugesan et al., 2021b; Ammanabrolu and Hausknecht, 2020; Wang et al., 2022b; Atzeni et al., 2021; Yao et al., 2020; Tuyls et al., 2021) in benchmark environments for text-based reinforcement learning such as Textworld (Côté et al., 2018), Jericho (Hausknecht et al., 2019), Scienceworld (Wang et al., 2022b), etc.

3 Results

We now present our main results. In the TWC environment, agents are trained for 100 episodes, with a maximum of 50 steps per episode (repeated over 5 runs). In the Jericho environment, agents were trained over 100000 steps with no limit to the number of episodes (repeated over 3 runs). These settings were chosen following previous work reference in this manuscript, such as Yao et al. (2021) and Murugesan et al. (2021b). Note that we report results for the game *Zork 1* in this section, the results observed here extend to other Jericho games and agent architectures as seen in Appendix A.6.

We deploy agents of the same architecture as described in Section 2, the only exception being that the input encoder used by them is different. The encoders are the *Hash* encoder, which produces semantic-less vectors, the Word *Embedding* which uses pre-trained GloVe embeddings, and the transformer LMs Albert (Lan et al., 2019) and

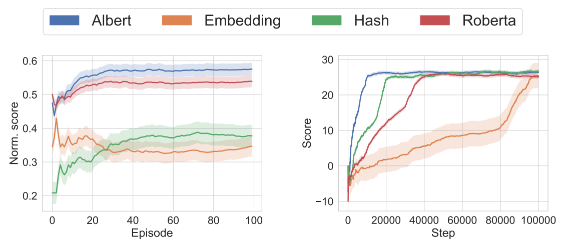


Figure 2: Training performance comparing various encoders; (left) shows the normalized scores for TWC games and (right) training scores in *Zork 1*. Shaded area corresponds to one standard deviation.

Model	ID	OOD
Hash	0.58 ± 0.06	0.15 ± 0.03
Embedding	0.58 ± 0.08	0.43 ± 0.07
Albert (Lan et al., 2019)*	0.66 ± 0.05	$\mathbf{0.65} \pm 0.05$
RoBERTa (Liu et al., 2019)*	$\mathbf{0.70} \pm 0.05$	0.53 ± 0.06

Table 1: Normalized scores for the in-distribution vocabulary (ID) and out-of-distribution vocabulary (OOD) game sets in TWC's Medium difficulty games. (*) Indicates fixed language models.

RoBERTa (Liu et al., 2019). The transformer encoders are used in two variations: *Fixed*, where the LMs weights are frozen; and *Fine-tuned (FT)*, where the LMs weights are updated according to the rewards. This allows us to compare the performance of the typical text-based RL fine-tuning approach to unconventional ones.

3.1 Semantic Information from Pre-Training Improves the Overall RL Performance

We evaluate the use of different LMs to encode the observations and actions into fixed-length vectors. We begin our analysis with the weights of the language model-based encoders fixed, i.e., only the RL network parameters ϕ are updated.

The rich semantic information of LMs accelerates training: The results from these experiments show that even an agent without semantic information can properly learn to play the games. However, an agent leveraging the semantic representations from language models are able to: (1) converge more quickly, in training, to a stable score than hash and simple, as shown in Figure 2; (2) handle out-of-training vocabulary, Table 1 shows the performance of the models under two settings: games using an in-training vocabulary (ID) and games using an out-of-training vocabulary (OOD). These results show that the fixed transformer LMs outperform the Hash and Embedding models in both vocabulary distributions, highlighting the importance of keeping the semantic information from pre-training intact.

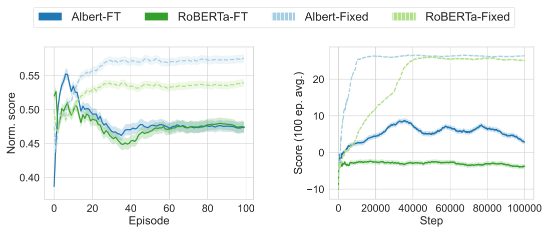


Figure 3: Training curves of fixed/fine-tuned LMs on (left) TWC medium difficulty games and (right) *Zork 1*. Due to semantic degeneration, the fine-tuned models do not exhibit an increasing score converging to a maximum value. Shaded areas denote one standard deviation.

3.2 Semantic Degeneration Hurts Learning

In this experiment, we address the first proposed research question: "does fine-tuning the LM to the RL rewards improve the training efficiency in comparison to fixed pre-trained LMs?" To that end, we trained the Fixed and Fine-tuned variations of Albert and RoBERTa encoders on the same games, and compared their scores during training. Figure 3 shows the outcome of the experiment. The findings suggest that traditional text-based RL approach of fine-tuning the LMs lead to substantially lower training scores, which are due to semantic degeneration. That is, **semantic degeneration leads to ineffective training of the RL agents**, whereas the fixed models converge to a higher score after a relatively small number episodes/steps.

Semantic degeneration arises from fine-tuning the LMs to the training rewards. The LM "forgets" its semantic associations it had learning during its pre-training, such as the masked token prediction in the case of transformers. This "forgetting" originates from overfitting the model's weights to the games' word distributions. The biggest problem arises from the fact that the RL network receives the encoded vectors from the LM and updates its weights based on such initial representations. However, since the LMs are fine-tuned, the encoding will change between each episode, causing the RL network to receive a different encoding for the same observation as the training goes on.

A comparison of pre-trained and semantically degenerated word vectors is seen in Figure 4. A 2D TSNE plot of pre-trained word vectors from a RoBERTa model is seen in Figure 4a; Figure 4b shows the plot of the word vectors after fine-tuning

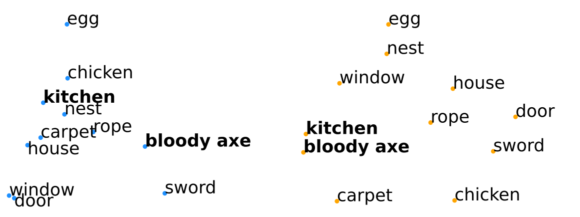


Figure 4: Shift caused by the semantic degeneration to the contextual word vectors in the RoBERTa model fine-tuned to *Zork 1*: (a) pre-trained embeddings, (b) embeddings fine-tuned to *Zork 1*. Bolded words denote the case where the term "bloody axe" shifts towards the word "kitchen" as a result of them co-occurring in a positively rewarded state.

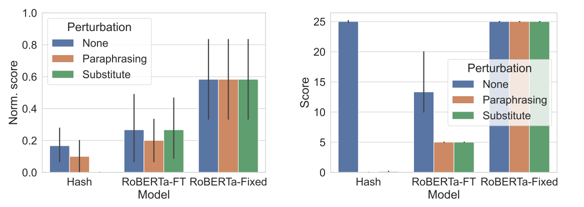


Figure 5: Evaluation of a RoBERTa agent on original (none), paraphrased, and lexical substitution observations on (left) TWC medium games and (right) *Zork 1*. Fixed LMs exhibit strong robustness to the perturbations, scoring as much as in the games without perturbations.

the LM to the game *Zork 1*. Notice the shift of the term "bloody axe" towards the term "kitchen" from (a) to (b). The shift happens because both terms appear in a sequence early on in the game, therefore, the association between their vectors becomes stronger as the LM is fine-tuned. Moreover, the terms "egg" and "nest" shift away from "chicken". The first two terms are also employed in the game in a sequence where the agent receives a positive reward, whereas the last is never used in the game. Despite being related *in-game*, these terms should have their semantic relationships preserved, which is possible by utilizing fixed LMs.

3.3 Agents with fine-tuned LMs are less robust to language change

We address the second research question: "does fine-tuning LMs make agents robust to tasks containing out-of-training vocabulary?". To test the robustness of each model, we first train each agent on a particular game. Then, we evaluate the agents by having them play games where the observations are transformed in one of the following ways: Paraphrasing, we run the observations through a paraphrasing model to rephrase the descriptions (using a Bart-based paraphrase (Lewis et al., 2019)); Lexical Substitution, we replace words in the observations using synonyms and hypernyms from WordNet (Fellbaum, 2010). By playing these versions of the games, agents have to perform the same task as seen in training, but with reworded or slightly modified observations.

Figure 5a shows the fixed LM agent is robust to paraphrasing as it is able to maintain the original score even in the modified versions. This is due to the ability of LMs to handle such perturbations in text. This evidence emphasizes the hypothesis that semantic understanding is important for generalization to words unseen in training. Figure 5b shows the performance of the three agents in *Zork 1*. The fine-tuned agent exhibits a decline in performance while playing the paraphrased and lexical substitution games. This is explained by the fact that the LM has been adjusted to the semantics of the original game, thus, tokens are no longer distributed according to semantic similarity. The hash-based agent is unable to score in either of the modified games due to the lack of semantic information. The fixed agent, however, exhibits strong robustness to the perturbations. This shows how semantic degeneration leads to decrease in performance in unseen or slightly different games.

4 Conclusion

In this paper, we have put forth a novel perspective over the occurrence of semantic degeneration at the intersection of LM fine-tuning and text RL. We have shown that semantic understanding brings benefits to the training of agents. Moreover, despite being the typical approach to text-based RL, learning the semantics from the game may not be the optimal approach to training agents. Our results corroborate the well known trends of trading-off general semantics for task-specific representations in NLP tasks; we shine light on how this affects agents in carrying out tasks that are semantically similar to the training ones. Semantic degeneration was observed for different agent architectures and in several games and environments. This suggests that the phenomenon is a general problem. Our results indicate that using meaningful semantic representations can be beneficial, and fine-tuning strategies may be developed to ensure prior semantic information is not lost by the model, while learning task-specific representations.

Limitations

Our work focuses on popular TBG environments and also popular choices of LMs. In future work it would be interesting to study rarer TBG environments, potentially beyond English. In that context it would also be interesting to study multilingual LMs as the semantic representation for these games. Since we use LM representations for game playing, some of the limitations of these representations (like inability to distinguish between some related concepts, or certain biases), might carry over. Investigating these in detail is another interesting avenue to be explored.

References

Prithviraj Ammanabrolu and Matthew Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces. *arXiv preprint arXiv:2001.08837*.

Mattia Atzeni, Shehzaad Zuzar Dhuliawala, Keerthiram Murugesan, and Mrinmaya Sachan. 2021. Case-based reasoning for better generalization in textual reinforcement learning. In *International Conference on Learning Representations*.

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*, pages 41–75. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Rahul Dey and Fathi M Salem. 2017. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE.

Christiane Fellbaum. 2010. Wordnet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer.

Matthew Hausknecht, Prithviraj Ammanabrolu, Côté Marc-Alexandre, and Yuan Xingdi. 2019. Interactive fiction games: A colossal adventure. *CoRR*, abs/1909.05398.

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*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. *arXiv preprint arXiv:2006.04884*.

Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Pushkar Shukla, Sadhana Kumaravel, Gerald Tesauro, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. 2021a. Text-based rl agents with commonsense knowledge: New challenges, environments and baselines.

Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. 2021b. Efficient text-based reinforcement learning by jointly leveraging state and commonsense graph representations. In *Acl-Ijcnlp 2021: The 59Th Annual Meeting Of The Association For Computational Linguistics And The 11Th International Joint Conference On Natural Language Processing, Vol 2*, pages 719–725. ASSOC COMPUTATIONAL LINGUISTICS-ACL.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Jens Tuyls, Shunyu Yao, Sham M Kakade, and Karthik R Narasimhan. 2021. Multi-stage episodic control for strategic exploration in text games. In *International Conference on Learning Representations*.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022a. Behavior cloned transformers are neurosymbolic reasoners. *arXiv preprint arXiv:2210.07382*.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022b. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11298.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. 2016. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*.

	Objects	Targets	Rooms
Easy	1	1	1
Medium	2–3	1–3	1
Hard	6–7	5–7	1–2

Table 2: No. of objects, target objects and rooms in TWC games per difficulty level.

Shunyu Yao, Karthik Narasimhan, and Matthew Hausknecht. 2021. Reading and acting while blindfolded: The need for semantics in text game agents. *arXiv preprint arXiv:2103.13552*.

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.

A Appendix

A.1 TextWorld Commonsense

This section contains information about the games (Table 2) in TextWorld Commonsense as well as an example of an observation and plausible actions (Figure 6).

The goal of TWC games are to complete a series of household tasks, such as "picking up an apple and putting it in an appropriate location". The agent is provided with the description of a scene and a list of plausible actions. They must then decide which action to be taken in the current game state. If the action performed is good, the agent is rewarded with points.

TWC games are split into easy, medium and hard difficulties. As the difficulty increases, the number of target objects and rooms to cleanup increases. Details can be seen in Table 2.

A.2 Model comparison in TWC

Figure 7 shows the comparison between all language models in all three difficulties of TWC in terms of normalized score and number of movements.

These results show how agents using fixed LMs converge earlier to a stable score (Figures 7 a, b, c) and to stable number of movements (Figures 7 d, e, f). Higher scores are better. Lower number of movements are better because it means the agent can complete the task while taking fewer actions, avoiding unnecessary moves.

A.3 Complete Table of TWC Results

Tables 4 and 5 show the results for all difficulties in TWC in the in-distribution set and out-of-distribution set.

We can see that fixed LMs consistently perform better when applied to both in-distribution and out-of-distribution tasks. This is due to the fact that they can keep rich semantic information and not suffering from semantic degeneration.

A.4 Complete results for perturbation experiments in TWC

Figure 8 shows the results for the perturbation experiments in TWC difficulties.

The result show how that a fixed LM model (RoBERTa) can maintain a relatively similar performance to the original observations when playing noisy versions of the game.

A.5 Experiment details

All experiments were repeated 3 times with different random seeds. The reported error bars correspond to one standard deviation. We followed previous literature for determining the hyperparameters for the agent model (Murugesan et al., 2021b; Yao et al., 2021).

For **TWC**, agents were trained for 500 episodes, each episode ended when the game was over or when the agent reached 100 steps. Results were reported as the normalized score for each game, on a scale between 0 and 1. The normalized score was calculated by dividing the final score by the maximum score possible for that game. We trained and evaluated the agents on all games of each of the three difficulties: easy, medium and hard.

For **Jericho**, agents were trained for 100,000 steps, regardless of the number of episodes this incurred in. The results were reported as the average score for the last 100 episodes played by the agent. We trained and evaluated agents on 12 Jericho games, as seen in Table 3.

A.6 Additional experiments on Jericho games

Figure 9 shows the fine-tuning/fixed LM comparison on additional games from the Jericho library: `detective`, `pentari`, `inhumane`, and `enchanter`. The models show a consistent trend in which the fixed LMs outperform the fine-tuned models.

Table 3 shows an extensive set of experiments on 12 Jericho games and different agent architectures. In addition to the Actor-Critic network de-

Observation
You've entered a kitchen.
Look over there! A dishwasher. You can see a closed cutlery drawer. You see a ladderback chair. On the ladderback chair you can make out a dirty whisk.

Plausible Actions
Open dishwasher
Open cutlery drawer
Take dirty whisk from ladderback chair

Figure 6: Example of an observation from a TextWorld Commonsense game.

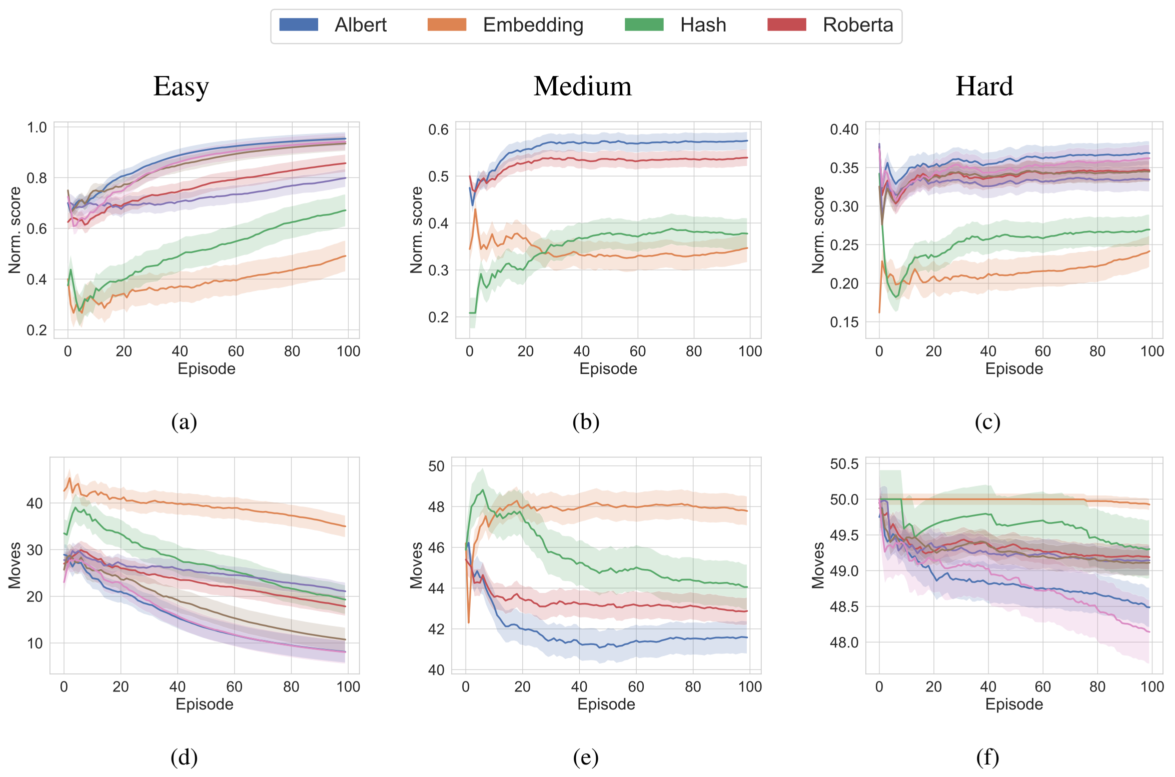


Figure 7: Comparison of the performance across several language encoding models. Figures a, b, c show the normalized score for easy, medium and hard games, respectively. Figures d, e, f show the number of movements needed by the agent to complete the task (lower values are better). Shaded region corresponds one standard deviation.

scribed in the manuscript, we conducted experiments with variants of the CALM agent (Yao et al., 2020), which utilizes a DRRN and a GPT generative model to score actions. In addition to the default CALM agent with DRRN, we introduced our LM combinations to it both with and without fine-tuning. We can see the trend is maintained for fine-tuned LMs regardless of the agent architecture, and regardless of the game environment.

A.7 Text perturbations

This sections presents a description of the perturbations applied to the game texts.

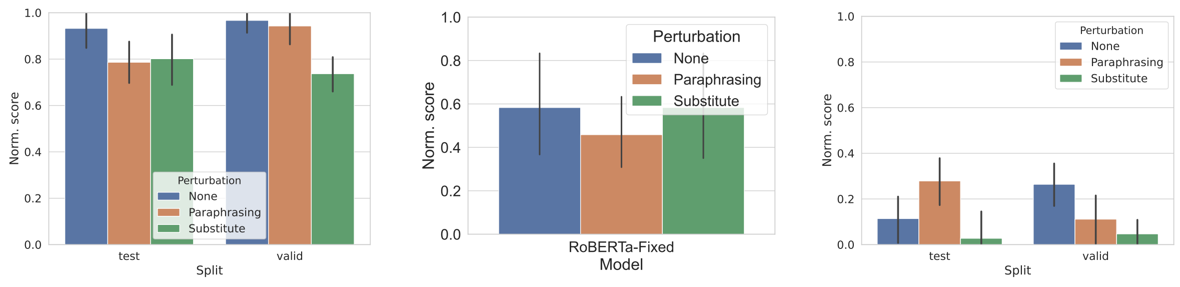
A perturbation is a modification of an original piece of text in the game to produce an "out-of-training" example. Perturbations are applied to the observations, actions and inventories.

The types of perturbations are:

- Lexical substitution - we use WordNet synsets to find replacements for words in the text
- Paraphrasing - we use a sequence-to-sequence BART paraphraser to rephrase the original text

B Reproducibility

The code needed used to implement the methods described in this manuscript are submitted along with the supplementary material. The code is anonymous and contains the instructions to set up the environments, download the game data, and train the agents.

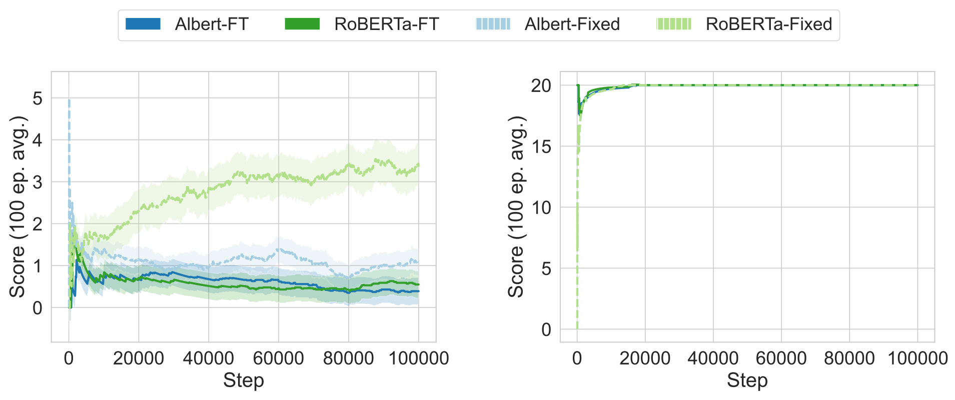


(a) TWC Easy (b) TWC Medium (c) TWC Hard

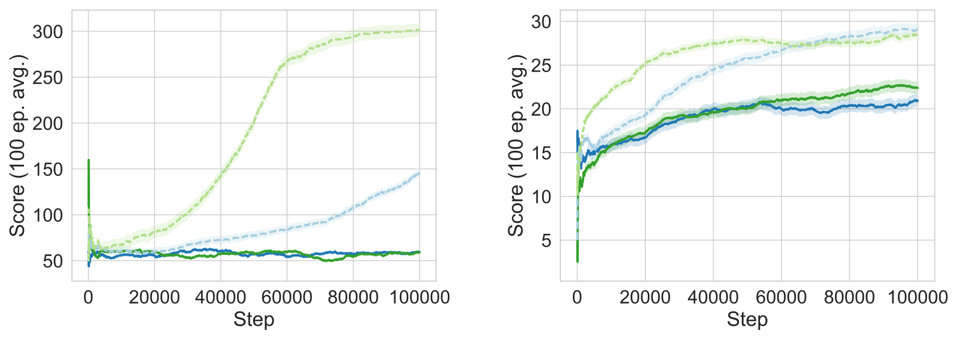
Figure 8: Evaluation of an RoBERTa agent on original, paraphrased, and lexical substitution observations on (a) Easy, (b) Medium and (c) Hard games.

Game	**Albert Fix.**	**Albert FT**	**RoBERTa Fix.**	**RoBERTa FT**	**Hash**	**CALM-DRRN**	**CALM Fix.**	**CALM-FT**
balances	10.0 ± 0.8	2.09 ± 1.01	10.0 ± 1.0	1.96 ± 0.92	9.0 ± 1.3	8.55 ± 1.1	9.02 ± 1.04	4.19 ± 1.93
deephome	7.0 ± 0.88	3.22 ± 2.01	7.0 ± 0.81	1.13 ± 1.1	3.0 ± 1.07	1.0 ± 1.02	4.0 ± 2.13	0.35 ± 0.5
detective	159.1 ± 2.03	58.7 ± 5.01	290.0 ± 10.0	57.9 ± 6.06	99.5 ± 8.1	290.0 ± 12.01	290.0 ± 11.3	122.84 ± 18.4
dragon	10.2 ± 3.22	-2.4 ± 5.01	11.0 ± 5.1	-5.04 ± 4.77	-6.31 ± 5.36	0.53 ± 3.7	6.05 ± 4.3	$0.40.53$
enchanter	18.3 ± 1.2	9.2 ± 2.01	20.0 ± 0.9	8.6 ± 1.38	18.0 ± 1.78	0.0 ± 0.0	8.84 ± 4.96	0.0 ± 0.0
inhumane	1.4 ± 0.54	0.63 ± 0.67	3.7 ± 0.33	0.67 ± 0.43	3.01 ± 0.39	12.7 ± 0.55	12.7 ± 0.58	8.13 ± 0.48
library	13.58 ± 0.73	10.93 ± 0.68	$12.990.83$	11.04 ± 0.71	12.53 ± 0.65	11.35 ± 0.59	11.35 ± 0.76	5.86 ± 0.77
ludicorp	12.64 ± 0.34	11.53 ± 0.42	12.13 ± 0.44	12.82 ± 0.47	12.99 ± 0.37	8.85 ± 0.83	8.9 ± 0.88	5.06 ± 0.78
omniquest	4.4 ± 1.2	1.01 ± 1.1	4.25 ± 1.22	1.75 ± 1.13	4.0 ± 2.0	5.95 ± 2.35	5.89 ± 2.76	4.35 ± 2.17
pentari	27.9 ± 0.33	17.25 ± 0.68	27.3 ± 0.28	22.0 ± 0.39	25.0 ± 0.30	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
zork1	28.6 ± 0.15	4.19 ± 0.71	26.92 ± 0.17	3.9 ± 0.91	24.0 ± 0.13	25.81 ± 0.15	26.02 ± 0.21	14.38 ± 0.29
zork3	0.01 ± 0.3	0.0 ± 0.0	0.02 ± 0.02	0.0 ± 0.0	0.01 ± 0.03	0.17 ± 0.05	0.2 ± 0.07	0.13 ± 0.3

Table 3: Evaluation of Fixed (Fix.) and Fine-tuned (FT) LMs across 12 games using the Actor-Critic and the CALM architectures. Values are the average scores of the last 100 episodes. Experiments were conducted 3 times for each model and game. Errors reported as one standard deviation.



(a) inhumane (b) enchanter



(c) detective (d) pentari

Figure 9: Comparison of fine-tuned/fixed LMs on various Jericho games.

	Easy		**Medium**		**Hard**	
Model	Score	Moves	Score	Moves	Score	Moves
DRRN	0.88 ± 0.04	24 ± 2	0.60 ± 0.02	44 ± 1	0.30 ± 0.02	50 ± 0
TPC	0.89 ± 0.06	21 ± 5	0.62 ± 0.03	43 ± 1	0.32 ± 0.04	48 ± 1
KG-A2C	0.86 ± 0.06	22 ± 3	0.62 ± 0.03	42 ± 0	0.32 ± 0.00	48 ± 1
BiKE	0.94 ± 0.00	18 ± 1	0.64 ± 0.02	39 ± 1	0.34 ± 0.00	47 ± 1
BiKE + CBR	0.95 ± 0.04	16 ± 1	0.67 ± 0.03	$\mathbf{35} \pm 1$	$\mathbf{0.42} \pm 0.04$	$\mathbf{45} \pm 1$
Hash	0.31 ± 0.07	43 ± 2	0.58 ± 0.06	43 ± 2	0.22 ± 0.03	50 ± 0
Simple	0.83 ± 0.08	26 ± 4	0.58 ± 0.08	43 ± 2	0.35 ± 0.05	49 ± 0
Albert*	0.96 ± 0.02	10 ± 2	0.66 ± 0.05	38 ± 2	0.41 ± 0.05	49 ± 0
MPNet*	0.85 ± 0.04	19 ± 3	0.66 ± 0.06	38 ± 2	0.36 ± 0.04	49 ± 0
RoBERTa*	0.94 ± 0.03	12 ± 2	$\mathbf{0.70} \pm 0.05$	38 ± 2	0.40 ± 0.04	49 ± 0
XLNet*	1.00 ± 0.00	$\mathbf{6} \pm 1$	0.65 ± 0.08	36 ± 3	0.37 ± 0.07	48 ± 1

Table 4: Results for the in-distribution (valid) sets in TWC. (*) Indicates agents with fixed LM encoders.

	Easy		**Medium**		**Hard**	
Model	Score	Moves	Score	Moves	Score	Moves
DRRN	0.78 ± 0.02	30 ± 3	0.55 ± 0.01	46 ± 0	0.20 ± 0.02	50 ± 0
TPC	0.78 ± 0.07	28 ± 4	0.58 ± 0.01	45 ± 2	0.19 ± 0.03	50 ± 0
KG-A2C	0.80 ± 0.07	28 ± 4	0.59 ± 0.01	43 ± 3	0.21 ± 0.00	50 ± 0
BiKE	0.83 ± 0.01	26 ± 2	0.61 ± 0.01	41 ± 2	0.23 ± 0.02	50 ± 0
BiKE + CBR	0.93 ± 0.03	17 ± 1	0.67 ± 0.03	35 ± 1	$\mathbf{0.40} \pm 0.03$	$\mathbf{46} \pm 1$
Simple	0.50 ± 0.12	39 ± 4	0.43 ± 0.07	43 ± 2	0.26 ± 0.04	50 ± 0
Hash	0.19 ± 0.06	44 ± 2	0.15 ± 0.03	50 ± 0	0.09 ± 0.02	50 ± 0
Albert*	0.64 ± 0.05	33 ± 3	0.65 ± 0.05	$\mathbf{38} \pm 2$	0.16 ± 0.02	50 ± 0
MPNet*	0.85 ± 0.05	23 ± 2	0.58 ± 0.06	42 ± 2	0.14 ± 0.02	50 ± 0
RoBERTa*	0.90 ± 0.04	19 ± 2	0.53 ± 0.06	44 ± 1	0.19 ± 0.03	50 ± 0
XLNet*	0.64 ± 0.05	30 ± 3	0.42 ± 0.07	47 ± 1	0.17 ± 0.03	50 ± 0

Table 5: Results for the out-of-distribution (test) sets in TWC. (*) Indicates agents with fixed LM encoders.