Towards Understanding Semantic Degeneration in 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 are able to 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 large language models (LLM) such as RoBERTa in guiding the agents. In this work, we describe the occurrence of semantic degeneration as a consequence of inappropriate fine-tuning language models in text-based reinforcement learning (TBRL). Our analysis shows that fine-tuning large language models may lead to poor performance. In addition, we show that, even though semantics is not required for successful training, a semantically rich representation improves the generalization of these agents.

1 Introduction

Text-based games (TBGs) are a form of interactive fiction applications where players use textual information to control characters and change the game's state in the environment. Due to the fact that the information from these games is shared as text, it is expected that a successful player exhibits 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 the conditional observation emission probabilities, and $R : S \times A \rightarrow \mathbb{R}$ the reward function. The goal of a TBG agent is to maximize the final game score by interacting with the environment through observed text and available actions.

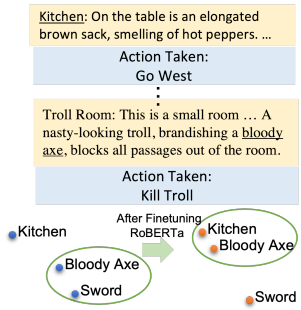


Figure 1: Semantic degeneration of the LMs after fine-tuning on trajectories from *Zork1*.

Recent works in TBRL adopt a strategy where semantics are learned from the game, typically by fine-tuning the underlying language models according to the rewards seen in training (Yao et al., 2020; Wang et al., 2022). 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 LLM pre-training, 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., 2021), and (2) Jericho (Hausknecht et al., 2019) to test our hypothesis. 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 it 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 apriori 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. Our results indicate that the semantic degeneration caused by finetuning LLMs to Q-values

during training leads to a decrease in the agent's performance and hinders its ability to play a different related game. Thus, we show that a better strategy now is to have agents learn the task at hand, without updating the semantic representations from the underlying LLM to preserve the rich semantic information from LLM pretraining. We hope that this work will help develop efficient finetuning strategies for interactive environments.

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 that takes as input the sequence of encoded observations 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.

Language Representation In order to assess the effectiveness of language models in text-based RL settings, we employ several models of distinct features. These models are used to encode the textual information, or observations, in the game into fixed-length vectors, which in turn are used as the input to the RL agent. During training, language models can be updated using the rewards from the game, or keep their weights frozen so as to avoid changing their word distributions. In particular, we use three types of encoders for our analysis:

- *Hash* (blindfolded) - this encoder does not capture semantic information from the text. Instead, we utilize a hash function to reduce the observation to an unique integer and use this number as the seed to generate a pseudo-random vector, similar to Yao et al. (2021).
- *Word embeddings* (simple) - we use static word embeddings (such as GloVe (Pennington et al., 2014)) to encode the tokens in the observations which are passed through a GRU to obtain a sequence encoding.
- *Transformer LLMs* - We use pre-trained LLMs to encode the observations (Devlin et al., 2018; Liu et al., 2019). These models have been shown to provide semantically rich encoding of textual information.

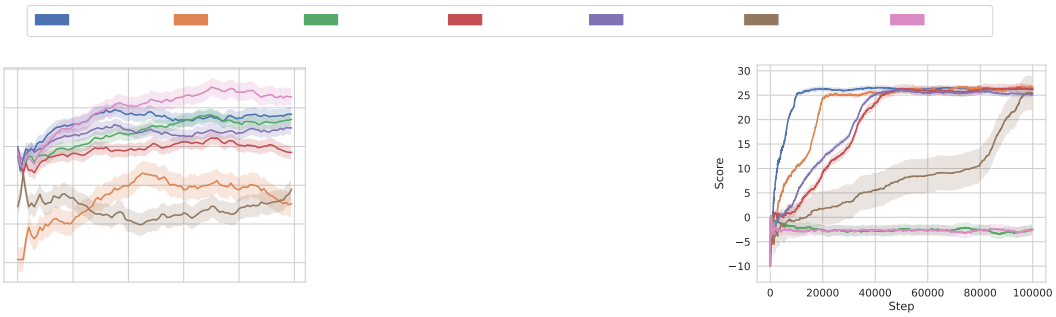
3 Experiments and Results

We now present our main experimental results. In the TWC environment, the agents are trained, unless otherwise stated, 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).

3.1 Comparing the performance of different LMs for input encoding

We evaluate the use of different LMs for encoding the textual observation and actions into fixed-length vectors. 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 range from static word embeddings (Simple) to LM-based models, and the blindfolded encoder (Hash). We begin our analaysis with the weights of the language model-based encoders fixed, i.e., only the agent's parameters ϕ are updated.

Semantic understanding makes learning more efficient in text-based games The results from these experiments show that even an agent without semantic information can properly learn to play the games, as seen in Figure 2. However, an agent leveraging the semantic representations from language models are able to: (1) converge to a stable score earlier and (2) generalize to unseen observations. Table 1 show that LM-based models using only text information match or outperform the baselines using sophisticated RL algorithms. We find that performance is consistent across the validation (in-distribution) and test (out-of-distribution) sets, which reinforce the argument that semantic understanding is key to generalization.



(a) Norm. scores (TWC) (b) Movements (TWC) (c) Scores (*Zork 1*)

Figure 2: Comparison of the performance across several LLM-based encoding models. Figures (a) and (b) show the normalized scores and number of movements (lower values are better) for the medium difficulty games in TWC. Figure (c) shows the game score achieved in training across 100k steps in *Zork 1*. Shaded area corresponds one standard deviation.

3.2 Comparison to baselines

We compare the performance of the fixed transformer models against baselines in the TWC and Jericho environments. In TWC, the performance of the models are measured in the in-distribution validation set, and in the out-of-distribution test set, which contains objects not seen during training, we report the average score and average number of moves taken for each data set. In Jericho, we compare the average score received in the last 100 episodes during training for 100,000 steps.

Evaluation and Analysis The results for the TWC environment in the medium difficulty are shown in Table 1. The top section of the table includes the baseline methods: DRRN (Hausknecht et al., 2019), TPC (Murugesan et al., 2021), KG-A2C (Ammanabrolu and Hausknecht, 2020), BiKE and BiKE + CBR (Atzeni et al., 2021). Our agent with LM encoders (e.g.: Albert, RoBERTa) for input encoding outperform baselines not only in the medium games but also in easy and hard (see Appendix A.3 for details). This result highlights the importance of semantic encoding, particularly in out-of-distribution sets, where there are objects not seen during training. In this case, the agent must rely on the language model to encode observations that might be similar to those seen in training even if the exact objects were not part of it.

3.3 Finetuning LMs with game trajectories and rewards

Next, we let the agent play the game for training and update the transformer weights in addition to the actor-critic network parameters. The results show that, under these settings, the agents hardly converge to a stable score. This happens because, when updating the language model, the semantic relationships between the words are broken as they are adapted to the current game's relations. We report the results for this experiment in Figure 3 using two of the best performing LMs in our set, Albert and RoBERTa.

Model	Valid		Test	
DRRN	0.60	0.02	0.55	0.01
TPC	0.62	0.03	0.58	0.01
KG-A2C	0.62	0.03	0.59	0.01
BiKE	0.64	0.02	0.61	0.01
BiKE + CBR	0.67	0.03	**0.67**	0.03
Hash	0.58	0.06	0.15	0.03
Simple	0.58	0.08	0.43	0.07
Albert*	0.66	0.05	0.65	0.05
MPNet*	0.66	0.06	0.58	0.06
RoBERTa*	**0.70**	0.05	0.53	0.06
XLNet*	0.65	0.08	0.42	0.07

Table 1: Results for the in-distribution (valid) and out-of-distribution (test) sets in TWC medium difficulty games. (*) Indicates fixed language models.

Finetuning the language representation to game trajectories and rewards causes semantic degeneration of the LMs The results presented in this section show how the agent's performance declines when language representations are updated as a result of training the reinforcement learning agent. We believe this behaviour is due to the overfitting of the representations to Q-values, leading to a degeneration of the semantic relationships learned during pre-training of LLMs. Moreover, finetuning the LM encoders to the game semantics seems to negatively impact the agent's ability to generalize as can be seen in the results from Section 3.4. Thus, it is desirable to avoid semantic degeneration altogether when training the agents by not updating the LMs weights directly.

A qualitative analysis of the semantic degeneration caused by finetuning LMs to Q-values can be seen in Figure 4. The figure show how the word

Figure 3: Performance of fine-tuned LLMs (Albert-ft, Roberta-ft) during training on TWC games in the easy difficulty. The degeneration of semantic understanding prevents the agent from converging to a stable score within 100 episodes, moreover, the normalized score exhibits a declining trend. Shaded areas denote one standard deviation.

vectors shift according to their co-occurrence in Zork 1. For instance, the vectors for *sword* and *bloody axe* are placed next to each other by the original model. The fine-tuned model, on the other hand, moves *bloody axe* closer to *kitchen* because these two entities co-occur in the neighboring states in a game trajectory that lead to positive rewards.

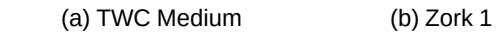
(a) (b)

Figure 4: Shift caused by the semantic degeneration to the contextual word embeddings in the RoBERTa model fine-tuned to Zork 1: (a) shows the word embeddings from the original model, (b) shows the word embeddings after fine-tuning to Zork 1. The bold words denote the case where the term "bloody axe" shifts towards the word "kitchen" as a result of these words appearing together in a state with positive reward.

3.4 Perturbations

Finally, we test the robustness of each model with respect to perturbations in the text. We evaluate the agents in 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 bart-based paraphrase); Lexical Substitution, we replace words in the observations using synonyms and hypernyms from WordNet (Fellbaum, 2010).

We see the performance of the fixed agent for TWC games in Figure 5a. In Easy and Medium difficulties, paraphrasing has little effect on the



(a) TWC Medium (b) Zork 1

Figure 5: Evaluation of a RoBERTa agent on original, paraphrased, and lexical substitution observations on (a) TWC medium games and (b) Zork 1. In Figure (a), we can see how each substitution renders the games more difficult for the agent as the observations contain out-of-training words. In Figure (b) we have a comparison of fixed, fine-tuned and hash language representations.

performance of the agent in both in-distribution (valid) and out-of-distribution (test) sets. This confirms the hypothesis that semantic understanding is important for generalization to "unseen" observations. 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 encoder has been fine-tuned to the semantics of the original game, thus, its encodings are no longer positioned according to semantic similarity. The hash-based agent is unable to score in either of the modified games. This is expected since the hash encoding does not capture any semantics. The fixed agent, however, exhibits strong robustness to the perturbations. This emphasizes the importance of using LLMs as-is to avoid degeneration of the rich semantic information.

4 Conclusion

In this paper, we have put forth a novel perspective where we leverage and adapt the semantic information contained in pre-trained language models to the task of playing text-based games using reinforcement learning. Our experiments show that agents using only fixed encodings of the textual features from transformer language models are able to leverage rich semantic information in the LMs to outperform baselines. Additionally, we show that updating the LM game semantics can negatively affect the semantic relationships learned by the pre-trained LMs, resulting in over fitted representations that do not exhibit transfer capabilities. For future work, we will focus on efficient methods to combine the rich semantic information of LMs with language information that are specific to the task.

Limitations

Our work focuses on popular TBG environments and also popular choices of LLMs. 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 LLMs as the semantic representation for these games. Since we use LLM 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 for future work.

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.

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.

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.

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.

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

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.

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

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.

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 dif culties. As the dif culty increases, the number of target objects and rooms to cleanup increases. Details can be seen in Table 2.

	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 dif culty level.

A.2 Model comparison in TWC

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

These results show how agents using xed 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 3 and 4 show the results for all dif culties in TWC in the in-distribution set and out-of-distribution set.

We can see that xed 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 dif culties.

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

A.5 Text perturbations

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

A perturbation is a modi cation 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 nd replacements for words in the text
- Paraphrasing - we use a 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.

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

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

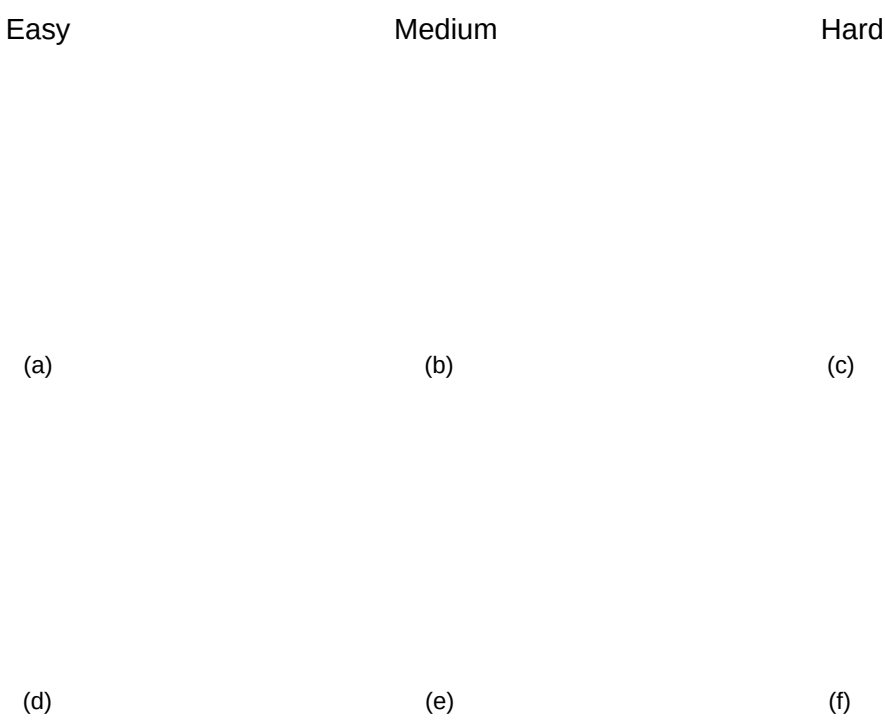


Figure 7: Comparison of the performance across several language encoding models. Figures a, b, c show the normalized score for easy, medium and hard games, respectively. Figures d, e, f show the number of movements needed by the agent to complete the task (lower values are better). Shaded region corresponds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

	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	35	1	0.42	0:04	45	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	0.70	0:05	38	2	0:40	0:04	49	0
XLNet*	1:00	0:00	6	1	0:65	0:08	36	3	0:37	0:07	48	1

Table 3: Results for the in-distribution (valid) sets in TWC. (*) Indicates agents with xed LM encoders.

Model	Easy		Medium		Hard	
	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 4: Results for the out-of-distribution (test) sets in TWC. (*) Indicates agents with fixed LM encoders.

(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.