Revisiting the Roles of "Text" in Text Games

Anonymous ACL submission

Abstract

Text games present opportunities for natural language understanding (NLU) methods to tackle reinforcement learning (RL) challenges. However, recent work has questioned the necessity of NLU by showing random text hashes could perform decently. In this paper, we pursue a fine-grained investigation into the roles of text in the face of different RL challenges, and reconcile that semantic and non-semantic language representations could be complementary rather than contrasting. Concretely, we propose a simple scheme to extract relevant contextual information into an approximate state hash as extra input for an RNN-based text agent. Such a lightweight plug-in achieves competitive performance with state-of-the-art text agents using advanced NLU techniques such as knowledge graph and passage retrieval, suggesting non-NLU methods might suffice to tackle the challenge of *partial observability*. However, if we remove RNN encoders and use approximate or even ground-truth state hash alone, the model performs miserably, which confirms the importance of semantic function approximation to tackle the challenge of *combinatorially large observation and action spaces*. Our findings and analysis provide new insights for designing better text game task setups and agents.

1 Introduction

In text-based games (Hausknecht et al., 2019; Côté et al., 2018), players command text actions to interact with a simulated world and gain rewards as they progress through the story (Figure 1). They can thus be seen as special partially observable Markov decision processes (POMDP), where observations and actions carry language semantics. Such a viewpoint motivates recent work to incorporate reinforcement learning (RL) agents with natural language understanding (NLU) capabilities for better performance. For example, pre-trained language models support generation in the *combinatorial action space* (Yao et al., 2020); commonsense

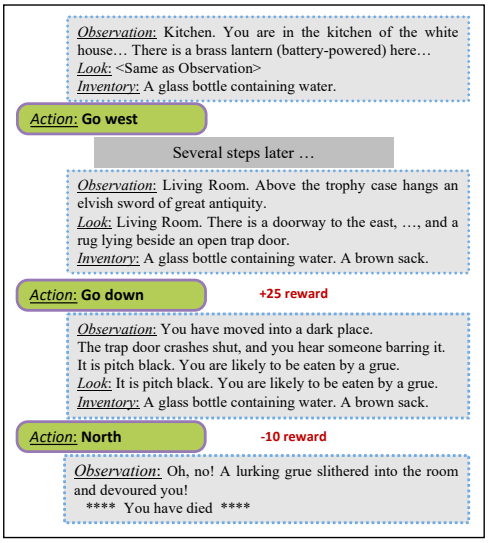

Figure 1: A game trajectory from Zork I.

reasoning (Murugesan et al., 2021), information extraction (Ammanabrolu and Hausknecht, 2020), and reading comprehension (Guo et al., 2020) systems provide priors for exploration with *sparse reward* and *long horizon*; and knowledge graph (Ammanabrolu and Hausknecht, 2020) and passage retrieval (Guo et al., 2020) techniques help alleviate *partial observability*.

Nevertheless, Yao et al. (2021) doubts the need of NLU for RL agents trained and evaluated on the same game. They found that a text game agent, DRRN (He et al., 2016), performs even slightly better when RNN-based language representations are replaced with non-semantic hash codes. Intuitively, hash serves to *memorize* state-action pairs and ignore text similarities, which is sometimes useful — consider the second-to-last observation in Figure 1 and a counterfactual observation where "A lantern" is added into "Inventory", RNNs might encode them very similarly though they lead to antipodal consequences (die or explore the underground). How do we reconcile this with recent NLU-augmented text agents with improved performances? Where are semantic representations useful, and where would a hash approach suffice?

In this paper, we present initial findings that semantic and non-semantic language representations could work hand-in-hand better than each alone by targeting different RL challenges. Concretely, we show the hash idea could help DRRN tackle *partial observability* – returning to the Figure 1 example, to get lantern to avoid death, it is vital to know where the lantern is, which is revealed in a previous instead of current observation. Based on such intuition, we propose a simple algorithm that tracks the current location and the up-to-date descriptions of all locations, then encode them into a single approximate state hash vector as extra DRRN input. Though lightweight and easy-to-implement, such a representation plug-in improves DRRN scores by 29% across games, with competitive performances against state-of-the-art text agents using advanced NLU techniques and pre-trained Transformer models. The effectiveness is further confirmed by comparing to models that plug in groundtruth state or location hash codes, where we find our performance and these upper bounds with very little gaps. These results suggest that the current *partial observability* bottlenecks might not require advanced NLU models or semantic representations to conquer.

However, such a message is gauged by the ablations that show the approximate state hash alone only achieves 58% of the full performance, as it fails to handle other RL challenges such as *the combinatorial state and action spaces*. In conclusion, we find the role of NLU in text games is not black-or-white as indicated by prior work, but rather differs for different RL challenges, and agents could benefit from combining semantic and non-semantic language representations that target different functionalities. Our results and insights contribute to future research in designing better tasks and models toward autonomous agents with grounded language abilities.

2 Preliminaries

2.1 Problem Formulation

A text game can be formulated as a partially observable Markov decision process (POMDP) $\langle S, A, T, O, \Omega, R, \gamma \rangle$, where at the t-th turn the agent reads a textual observation $o_t = \Omega(s_t) \in O$ as a partial reflection of underlying world state $s_t \in S$, issues a textual command $a_t \in A$ in response, and receives a sparse scalar reward $r_t = R(s_t, a_t)$ in light of game progress. The state transition $s_{t+1} = T(s_t, a_t)$ is hidden to the agent. The goal is to maximize the expected cumulative discounted rewards $\mathbf{E}[\sum_t \gamma^t r_t]$.

Observations and States Following prior practice in the Jericho benchmark (Hausknecht et al., 2019), we augment the direct observation o_t with inventory i_t and location description l_t obtained by issuing actions "*inventory*" and "*look*" respectively. But even this may not reveal the complete s_t (Section 1), which is Jericho includes an object tree and a large simulator RAM array hidden to players. As s_t is large and lacks interpretability, more often used is the state hash $\mathrm{h}(s_t)$, where $\mathrm{h} : S \to \mathbb{N}$ maps each state to an integer that can be used to probe if states are identical, but cannot provide *semantic* information about state differences. Access to s_t or $\mathrm{h}(s_t)$ is a handicap in Jericho.

2.2 The DRRN Baseline and its Hash Variant

Denote $c_t = (o_1, a_1, \cdots, o_t)$ as the game context up to o_t, and for convenience we omit the subscript t when no confusion is caused. Our baseline RL model, Deep Reinforcement Relevance Network (DRRN) (He et al., 2016), learns a Q-network

$$Q(c_t, a_t) = \mathrm{MLP}(sr, ar) \tag{1}$$

where the **state and action representations**

$$\begin{aligned} sr_{\text{drrn}} &= [\mathrm{GRU}_1(o_t), \mathrm{GRU}_2(i_t), \mathrm{GRU}_3(l_t)] \\ ar_{\text{drrn}} &= \mathrm{GRU}_4(a_t) \end{aligned} \tag{2}$$

are encoded by gated recurrent units (GRU) (Cho et al., 2014). The temporal difference (TD) loss and Boltzmann exploration are used for RL.

In Yao et al. (2021), (2) is replaced by random, fixed, non-semantic hash representations

$$\begin{aligned} sr_{\text{hash}} &= [H(o_t), H(i_t), H(l_t)] \\ ar_{\text{hash}} &= H(a_t) \end{aligned} \tag{3}$$

where a hash vector function $H = vec \circ h$ first maps inputs to integers (via Python built-in hash) then to random normal vectors (by using the integer as the generator seed). However, neither of the models addresses partial observability by using the context c_t beyond the current observation o_t.

3 Method

The key to handle partial observability is to extract the appropriate state-distinguishing information from the context c_t — while under-extraction leads to different states with same representations,

Game	*DRRN and Variants*				*Agents with advanced NLU*		Max
	DRRN	Obs Hash	+ Inv-Dy	LoG (ours)	MPRC-DQN	KG-A2C	
zork1	39.4/53	35.5/50	43.1/87	**51.2/107**	38.3/–	33.6/35	350
zork3	0.4/4.5	0.4/4	0.4/4	1.33/**5**	**3.0/5.0**	0.1/–	7
pentari	26.5/45	**51.9/60**	37.2/50	44.4/**60**	44.4/–	48.2/56	70
detective	290/**337**	290/317	290/323	288.8/313.3	**317.7**/–	246.1/274	360
ludicorp	12.7/23	14.8/23	13.5/23	16.7/23	10.9/**40.7**	**17.6**/19	150
inhumane	21.1/45	21.9/45	19.6/45	25.7/**56.7**	**29.8**/53.3	3/–	90
avg norm	.28/.52	.34/.52	.30/.51	.36/.59	.41/-	.27/-	

Table 1: Final episodic/maximum explored scores for different games. MPRC-DQN numbers with max scores correspond to version change of games, so we re-run their model and report the new results. Average normalized score (avg norm) is model score divided by maximum game score, averaged across games.

over-extraction leads to diverging representations for the same state with different history paths. So to approximate the state hash, we first obtain and maintain a location map by exploration with limited depth d, collecting names of adjacent rooms:

$$po_1 = \{(p, loc(c_t, p) \mid p \subset A^d\} \quad (4)$$

where p is a sequence of navigation actions, and loc is the location after following p from c_t. Essentially, $po_1(c_t)$ serves to distinguish different locations with same names (e.g. "maze" rooms with the same observation).

Secondly, we collect the most-recent location descriptions for all locations, so that we may know, for example, the whereabouts of the lantern when needed (Section 1).

$$po_2 = \{(loc, \text{LastLook}(loc)) \mid loc \in \text{Map}\} \quad (5)$$

Together, our model DRRN-WorldHash (WH) takes state representation

$$sr_{\text{WH}} = [sr_{\text{drrn}}, H(po_1)), H(po_2)] \quad (6)$$

The algorithm details are in Appendix A.

4 Experiments

Implementation Details We adopt DRRN hyperparameters from Yao et al. (2021) to train our model. Following previous work, we implement the BiDAF (Seo et al., 2016) attention mechanism and the inverse dynamics auxiliary objective (Yao et al., 2021) for better text encoding. The episodic limit is 100 steps and the training has 1,000 episodes from 8 parallel game environments. For po_1, we use $d = 1$ as depth limit. We train three independent runs for each game. More details are in Appendix B.

Baselines Our approach builds on the backbone DRRN agent, thus we provide apple-to-apple comparisons to the original DRRN and its hash and inverse dynamics variants from Yao et al. (2021). We also compare with more complex state-of-the-art agents that are designed to deal with the partial observability via NLU:

- **MPRC-DQN** (Guo et al., 2020), which retrieves the relevant history to enhance the current observation, and formulates the action prediction as a multi-passage reading comprehension problem.
- **KG-A2C** (Ammanabrolu and Hausknecht, 2020; Ammanabrolu et al., 2020), which extracts an object graph with OpenIE (Angeli et al., 2015) or a BERT-based QA model (Devlin et al., 2019), and embeds the graph to a single vector as the state representation. We compare with the better result from the two papers for each game.

Evaluating Games We select 6 games from Jericho (Hausknecht et al., 2019) where **MPRC-DQN** or **KG-A2C** exhibits performance boosts, thus are more likely to suffer from partial observability.

4.1 Game Results

Table 1 shows game scores for all models. Among DRRN and it variants, DRRN-WH performs best on 4 of the 6 games. More impressively, our agent is competitive against MPRC-DQN (better or same score on 3/6 games) and KG-A2C (better scores on 4/6 games) in terms of winning rates. Overall, our DRRN-WH achieves the second best average normalized score of 36%, only behind 41% of MPRC-DQN (which is largely attributed to Zork III). Considering the fact that we explicitly choose the six games in favor of these two state-of-the-art baselines, such a result indicates that advanced NLU techniques might not be a must to solve partial observability — at least in the scoring ranges of current text game agents (i.e. average normalized score less then 50%).

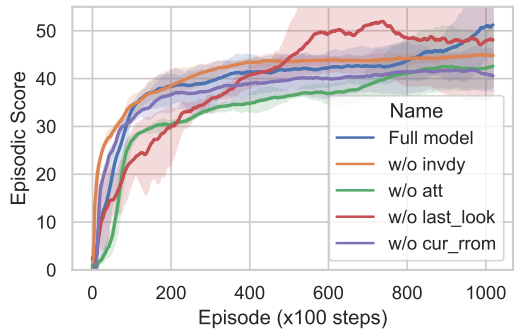

Figure 2: Ablation results on Zork I.

4.2 Upper Bound Analysis with Oracle States

Next, we aim to upper bound our model by replacing the approximate state hash with the groundtruth state hash (GT-State) from Jericho, which could perfectly distinguish different states apart, where $sr_{gt} = (sr_{\text{drrn}}, H(s))$.

As shown in Table 2, the scores of DRRN-WH and the GT-State are very close across different games, meaning our approximation has been close-to-perfect within the state hashing scheme. Notably, even GT-State fails to totally surpass **MPRC-DQN** or **KG-A2C**, suggesting NLU techniques might help these agents with RL challenges other than partial observability. Finally, we also show in Appendix C the performance of replacing our state approximation with the groundtruth room IDs (GT-Room), where our agent achieves on-par or better results on all the games. This confirms that our state approximation not only effectively identifies player locations by (4), but also brings richer state information thanks to (5).

4.3 Ablation Studies

Does the good performance of DRRN-WH indicate that distinguishing states and memorizing trajectories are all it takes to solve one text game? To answer this question, we conduct ablation experiments to remove the text encoder (i.e. all GRUs) in our agent as well as the GT-State version (**- Text Enc**). Intuitively, this renders the text game into a large, deterministic MDP (instead of POMDP), where even very close states (e.g. same except a window is ajar or open) have completely different representations.

The result in shown in Table 2, which witnesses a huge performance drop for DRRN-WH and its GT-State version with text encoders removed — in other words, learning the text game as a tabular MDP without language semantics could lead to a much deteriorated sample complexity, even

Game	Ours		Ours w/ GT-State	
	Full Model	- Text Enc.	Full Model	- Text Enc.
zork1	51.2/107	4.13/36.3	53.6/111	6.25/39.3
zork3	1.33/5	0.85/3	1.50/4.7	1.02/4
pentari	44.4/60	20.3/45	46.1/60	20/45
detective	288.8/313.3	281.3/313.3	289.9/310	280/290
ludicorp	16.7/23	10.15/22	15.9/23	9.5/21
inhumane	25.7/56.7	1.9/23	24.1/60	1.1/20
Avg. Norm	.36/.59	.21/.40	.37/.59	.22/.42

Table 2: The results of replacing our state representations with groundtruth state IDs (GT-State Full Model), as well as removing the text encoder (- Text Enc).

when partial observability is solved. To explain why DRRN with GT-State hash is much worse than DRRN with observation hash proposed in Yao et al. (2021), note that (3) still leverages the compositional structure of (o_t, i_t, l_t), e.g. two states with the same i_t still share part of the state representation. Such a result helps confirm the importance of language for the RL challenge of **large observation and actions spaces**: semantics-preserving function approximation (e.g. RNN instead of hash) could be key to interpolation (smooth value estimation for similar states) as well as extrapolation (efficient exploration based on language and commonsense priors).

Finally, we ablate individual components of DRRN-WH on Zork I. Figure 2 shows that removing the language-learning auxiliary task of inverse dynamics (**w/o invdy**) or the language attention (**w/o att**) leads to worse scores, reconfirming that semantic language representations are vital for DRRN-WH's success. On the other hand, removing the current whereabouts (**w/o cur_room**) leads to much worse performance than removing location descriptions across the map (**w/o last_look**), suggesting location identification (4) might be more important for solving partial observability.

5 Discussion

We propose a simple approach to deal with partial observability in text games, which could serve as a competitive baseline for future research, and also inspire similar investigations for other RL challenges to test the limits of memorization and necessity of NLU in different dimensions, which would in turn help identify flaws of current setups and propose better ones. We also hope our idea of best combing semantic and non-semantic language representations could be useful for building next-generation text game agents, as well as for other language applications with memorization needs like closed-domain QA or goal-oriented dialog.

References

Prithviraj Ammanabrolu and Matthew Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces. *arXiv*, pages arXiv–2001.

Prithviraj Ammanabrolu, Ethan Tien, Matthew Hausknecht, and Mark O Riedl. 2020. How to avoid being eaten by a grue: Structured exploration strategies for textual worlds. *arXiv preprint arXiv:2006.07409*.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186.

Xiaoxiao Guo, Mo Yu, Yupeng Gao, Chuang Gan, Murray Campbell, and Shiyu Chang. 2020. Interactive fiction game playing as multi-paragraph reading comprehension with reinforcement learning. *arXiv preprint arXiv:2010.02386*.

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

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630.

Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. 2021. Efficient text-based reinforcement learning by jointly leveraging state and commonsense graph representations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 719–725.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

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.

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 Algorithm of Our Approximated State Representation Construction

Algorithm 1 Infer the current location with nearby room names using depth-first search with limited depth. This can help distinguish different rooms with the same name in most cases. We use $depth = 1$ in our runs.

```
1:  function LOCATE(env, depth)
2:      state ← current state of env
3:      room ← current room name in env
4:      nearby ← []
5:      for all direction do
6:          step env with action direction
7:          if env changed in last action then
8:              if depth > 1 then
9:                  d ← LOCATE(env, depth − 1)
10:             else
11:                 d ← current room name in env
12:             append (direction, d) to nearby
13:             set env with state
14:     return nearby
```

Algorithm 2 Maintain the state approximation with the descriptions the last time we visit each room. The result is then hashed to serve as the *state hash* in our model. For runs provided with grounded room ID, we replace line 3 with the ground truth.

```
1: state ← {}
2: function UPDATEANDGETSTATE(env)
3:     room ← LOCATE(env, depth)
4:     look ← look of the current state in env
5:     state[room] ← look
6:     return state
```

B More Details of Our Model and Implementation

We follow the hyperparameters from Yao et al. (2021). For the state approximation part, we use the builtin HASH function in Python. We train our model for 10^5 steps, which takes about 40 hours on a TITAN X or Geforce GTX 1080.

We use the latest Jericho version 3.1.0. Due to a bug in Zork I, we add a timeout in the library to filter out valid actions causing the emulator to hang.

Game	Ours		Ours w/ GT-State		Ours w/
	Full Model	- Text Enc.	Full Model	- Text Enc.	GT-Room
zork1	51.2/107	4.13/36.3	53.6/111	6.25/39.3	52.0/110
zork3	1.33/5	0.85/3	1.50/4.7	1.02/4	1.31/5
pentari	44.4/60	20.3/45	46.1/60	20/45	44.8/58
detective	288.8/313.3	281.3/313.3	289.9/310	280/290	289.6/300
ludicorp	16.7/23	10.15/22	15.9/23	9.5/21	16.9/23
inhumane	25.7/56.7	1.9/23	24.1/60	1.1/20	25.7/56.7
Avg. Norm	.36/.59	.21/.40	.37/.59	.22/.42	.36/.58

Table 3: The results of replacing our state representations with groundtruth state IDs (GT-State Full Model), as well as removing the text encoder (- Text Enc).

B.1 Details of Our BiDAF Observation Encoder

In DRRN, the GRU takes the responsibility of both memorizing the high-scoring trajectories, and generalizing to unseen observations. In our method, the memorization power can be provided with our hash codes of local graphs, with stronger ability to distinguish states. We thus hope to encourage the generalization strength of neural network; and propose the attentive extension of observation embedding.

Our key idea bases on the insight that the Q-value in DRRN is computed by matching the textual observations to a textual action. Since the observations are usually significantly longer than the actions, the effect of an action can usually be determined by its interaction with a local context in the observation. This can be naturally modeled with the attention mechanism. Specifically, we apply the BiDAF (Seo et al., 2016) to match each observation component to the action.

The BiDAF takes the observation and action embeddings; and outputs an action-attended observation embedding. We denote the GRU embeddings for observation word i and action word j as $\boldsymbol{o}_i$ and $\boldsymbol{a}_j$. The attention score from an observation word to an action word is thus $\alpha_{ij}=\exp(a_{ij})/\sum_j \exp(a_{ij})$, where $a_{ij}=\boldsymbol{o}_i^T \boldsymbol{a}_j$. We then compute the "action-to-observation" summary vector for the i-th observation word as $\boldsymbol{c_i}=\sum_j \alpha_{ij}\boldsymbol{a_j}$. We concatenate and project the output vectors as $\left[\boldsymbol{o_i}, \boldsymbol{c_i}, \boldsymbol{o_i} \odot \boldsymbol{c_i}, |\boldsymbol{o}_i - \boldsymbol{c_i}|\right]$, followed by a linear layer with leaky ReLU activation units. We apply the aforementioned steps to the inventory i and location appearance l, too. Finally, we have

$$sr_{\text{LoG}} = [\text{BiDAF}(\boldsymbol{o}, \boldsymbol{a}), \text{BiDAF}(\boldsymbol{i}, \boldsymbol{a}), \text{BiDAF}(\boldsymbol{l}, \boldsymbol{a}), \text{H}(po_1(c))), \text{H}(po_2(po_1(c))] \quad (7)$$

C Additional Experiments with Oracle State Information

We investigate of performance of replacing our state approximation with the groundtruth room IDs (GT-Room). The results show that our agent achieves on-par or better results on all the games. This confirms that our state approximation not only identifies the true location of the player, but also brings richer state information.