Interactive Fiction Game Playing as Multi-Paragraph Reading Comprehension with Reinforcement Learning

Xiaoxiao Guo*
IBM Research
`xiaoxiao.guo@ibm.com`

Mo Yu*
IBM Research
`yum@us.ibm.com`

Yupeng Gao
IBM Research
`yupeng.gao@ibm.com`

Chuang Gan
MIT-IBM Watson AI Lab
`chuangg@ibm.com`

Murray Campbell
IBM Research
`mcam@us.ibm.com`

Shiyu Chang
MIT-IBM Watson AI Lab
`shiyu.chang@ibm.com`

Abstract

Interactive Fiction (IF) games with real human-written natural language texts provide a new natural evaluation for language understanding techniques. IF games pose language understanding challenges on the human-written textual descriptions of diverse and sophisticated game worlds and language generation challenges on the action command generation from less restricted combinatorial space. We take a novel perspective of IF game solving and re-formulate it as Multi-Passage Reading Comprehension (MPRC) tasks. Our approaches utilize the context-query attention mechanisms and the structured prediction in MPRC to efficiently generate and evaluate action outputs and apply an object-centric historical observation retrieval strategy to mitigate the partial observability of the textual observations. Extensive experiments on the recent IF benchmark (*Jericho*) demonstrate clear advantages of our approaches achieving high winning rates. [2]

1 Introduction

The complexity of Interactive Fiction (IF) games demands more sophisticated Natural Language Understanding (NLU) techniques than those used in synthetic text games. Moreover, the task of designing IF game-play agents, intersecting NLU and reinforcement learning (RL), poses several unique challenges on the NLU techniques. The first challenge is the difficulty of exploration in ***the huge natural language action space***. Previous approaches, starting with a single embedding vector of the observation, either predict the elements of actions independently [12, 7]; or embed each valid action as another vector and predict action value based on the vector-space similarities [9]. These methods do not consider the compositionality or role-differences of the action elements, or the interactions among them and the observation.

The second challenge is ***partial observability***. The latest observation is often not a sufficient summary of the interaction history and may not provide enough information to determine the long-term effects of actions. Previous approaches address this problem by building a representation over past observations (e.g., building a graph of objects, positions, and spatial relations) [3, 2]. These methods treat the historical observations equally and summarize the information into a single vector without focusing on important contexts related to the action prediction for the current observation.

* Equal contribution from the corresponding authors.
[2] Source code is available at: `https://github.com/XiaoxiaoGuo/rcdqn`.

Wordplay: When Language Meets Games Workshop @ NeurIPS 2020,
`https://wordplay-workshop.github.io/`

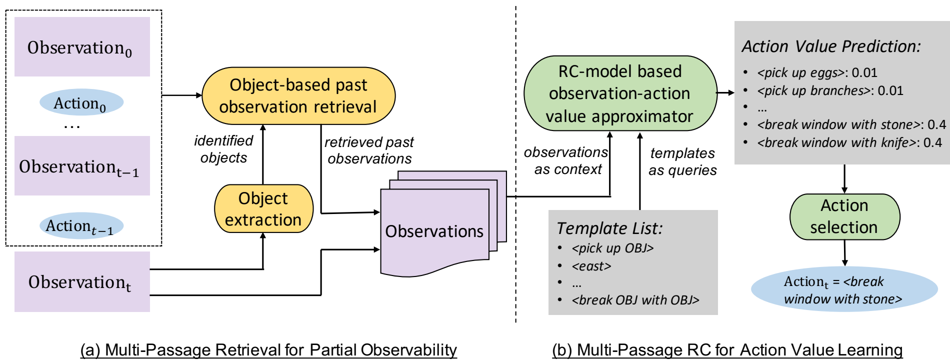

Figure 1: Overview of our approach to solving the IF games as multi-paragraph reading comprehension (MPRC) tasks.

We propose a novel formulation of IF game playing as Multi-Passage Reading Comprehension (MPRC) and harness MPRC techniques to solve the *huge action space* and *partial observability* challenges. The graphical illustration is shown in Figure 1. First, the action value prediction (i.e., predicting the long-term rewards of taking an action) is essentially *generating and scoring a compositional action structure by finding supporting evidence from the observation*. We base on the fact that each action is an instantiation of a **template**, i.e., a verb phrase with a few placeholders of object arguments it takes (Figure 1b). Then the action generation process can be viewed as extracting objects for a template's placeholders from the textual observation, based on the interaction between the template verb phrase and the relevant context of the objects in the observation. Our approach addresses the structured prediction and interaction problems with the idea of context-question attention mechanism in RC models. Specifically, we treat the observation as a passage and each template verb phrase as a question. The filling of object placeholders in the template thus becomes an extractive QA problem that selects objects from the observation given the template. Simultaneously each action (i.e., a template with all placeholder replaced) gets its evaluation value predicted by the RC model. Our formulation and approach better capture the fine-grained interactions between observation texts and structural actions, in contrast to previous approaches that represent the observation as a single vector and ignore the fine-grained dependency among action elements.

Second, alleviating partial observability is essentially *enhancing the current observation with potentially relevant history* and *predicting actions over the enhanced observation*. Our approach retrieves potentially relevant historical observations with an object-centric approach (Figure 1a), so that the retrieved ones are more likely to be connected to the current observation as they describe at least one shared interactive object. Our attention mechanisms are then applied across the retrieved multiple observation texts to focus on informative contexts for action value prediction.

We evaluated our approach on the suite of *Jericho* IF games, compared to all previous approaches. Our approaches achieved or outperformed the state-of-the-art performance on **20** out of 28 games.

2 Related Work

IF Game Agents. Previous work mainly studies the text understanding and generation in parser-based or rule-based text game tasks, such as TextWorld platform [6] or custom domains [12, 9, 1]. The recent platform *Jericho* [7] supports over thirty human-written IF games. Earlier successes in real IF games mainly rely on heuristics without learning. NAIL [8] is the state-of-the-art among these "no-learning" agents, employing a series of reliable heuristics for exploring the game, interacting with objects, and building an internal representation of the game world. With the development of learning environments like *Jericho*, the RL-based agents have started to achieve dominating performance.

A critical challenge for learning-based agents is how to handle the **combinatorial action space** in IF games. LSTM-DQN [12] was proposed to generate verb-object action with pre-defined sets of possible verbs and objects, but treat the selection and learning of verbs and objects independently. Template-DQN [7] extended LSTM-DQN for template-based action generation, introducing one additional but still independent prediction output for the second object in the template. Deep

Reinforcement Relevance Network (DRRN) [9] projects each action into a hidden space that matches the current state representation vector for action selection. Action-Elimination Deep Q-Network (AE-DQN) [15] learns to eliminate invalid action for efficient policy learning via utilizing expert demonstration data.

Other techniques focus on addressing the **partial observability** in text games. Knowledge Graph DQN (KG-DQN) [3] constructs and represents the game states as knowledge graphs with objects as nodes and uses pre-trained general purposed OpenIE tool and human-written rules. KG-DQN handles the action representation following DRRN. KG-A2C [2] later extends the work for IF games, by adding additional information extraction heuristics to fit the complexity of the object relations in IF games and utilizing a GRU-based action generator to handle the action space.

3 Multi-Paragraph Reading Comprehension for IF Games

Problem Formulation. Each IF game can be defined as a Partially Observable Markov Decision Process (POMDP), namely a 7-tuple of $\langle\ S,\ A,\ T,\ O,\ \Omega,\ R,\ \gamma\ \rangle$. , representing the hidden game state set, the action set, the state transition function, the set of textual observations composed from vocabulary words, the textual observation function, the reward function, and the discount factor respectively. The game playing agent interacts with the game engine in multiple turns until the game is over or the maximum number of steps is reached. At the t-th turn, the agent receives a textual observation describing the current game state $o_t \in O$ and sends a textual action command $a_t \in A$ back. The agent receives additional reward scalar r_t which encodes the game designers' objective of game progress. Thus the task of the game playing can be formulated to generate a textual action command per step as to maximize the expected cumulative discounted rewards $\mathbf{E}\Big[\sum_{t=0}^{\infty}\gamma^t r_t\Big]$. Value-based RL approaches learn to approximate a state-action value function $Q(o_t, a_t; \boldsymbol{\theta})$ which measures the expected cumulative rewards of taking action a_t when observing o_t. The agent selects action based on the action value prediction of $Q(o, a; \boldsymbol{\theta})$.

Template Action Space. Template action space considers actions satisfying decomposition in the form of $\langle$*verb*, *arg*$_0$, *arg*$_1\rangle$. For example, the action command [east], [pick up eggs] and [break window with stone] can be represented as template actions $\langle$*east*, *none*, *none*$\rangle$, $\langle$*pick up OBJ*, *eggs*, *none* and $\langle$*break OBJ with OBJ*, *window*, *stone*$\rangle$. We re-use the template library and object list from *Jericho*, which are extracted from human game play records. The verb phrases *verb* usually consist of several vocabulary words and each object *arg*$_{0/1}$ is usually a single word.

RC-based Template Action Prediction. We parameterize the observation action value function $Q(o, a{=}\langle verb, arg_0, arg_1\rangle; \boldsymbol{\theta})$ by utilizing the decomposition of the template actions and context-query contextualized representation in RC. Our model treats the observation o as a context in RC and the *verb*$=(v_1, v_2, ..., v_k)$ component of the template actions as a query. Then a *verb*-aware observation representation is derived via a RC reader model with Bidirectional Attention Flow (BiDAF) [14] and self-attention. The observation representation responding to the *arg*$_0$ and *arg*$_1$ words are pooled and projected to a scalar value estimate for $Q(o, a{=}\langle verb, arg_0, arg_1\rangle; \boldsymbol{\theta})$. A high-level model architecture of our model is illustrated in Figure 2.

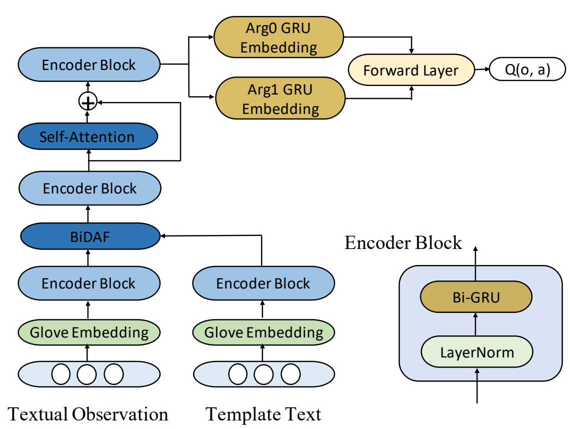

Figure 2: Our RC-based action prediction model architecture. The model details are in Appendix A.

Multi-Paragraph Confidence Method for Partial Observations We propose to retrieve past observations with an object-centric approach. Multiple past observations may share objects with

Game	Walkthrough-100	TDQN	DRRN	KG-A2C	MPRC-DQN	RC-DQN
acorncourt	30	1.6	**10**	0.3	**10.0**	**10.0**
advent	113	36	36	36	**63.9**	36
adventureland	42	0	20.6	0	**24.2**	21.7
afflicted	75	1.3	2.6	–	**8.0**	**8.0**
balances	30	4.8	**10**	**10**	**10**	**10**
detective	350	169	197.8	207.9	**317.7**	291.3
dragon	25	-5.3	-3.5	0	0.04	**4.84**
enchanter	125	8.6	**20**	12.1	**20.0**	**20.0**
gold	30	**4.1**	0	–	0	0
inhumane	70	0.7	0	**3**	0	0
jewel	24	0	1.6	1.8	**4.46**	2.0
karn	40	0.7	2.1	0	**10.0**	**10.0**
library	30	6.3	17	14.3	17.7	**18.1**
ludicorp	37	6	13.8	17.8	**19.7**	17.0
omniquest	50	**16.8**	10	3	10.0	10.0
pentari	60	17.4	27.2	**50.7**	44.4	43.8
reverb	50	0.3	**8.2**	–	2.0	2.0
snacktime	50	**9.7**	0	0	0	0
sorcerer	150	5	20.8	5.8	**38.6**	38.3
spellbrkr	160	18.7	**37.8**	21.3	25	25
spirit	8	0.6	0.8	1.3	3.8	**5.2**
temple	20	7.9	7.4	7.6	**8.0**	**8.0**
tryst205	50	0	9.6	–	**10.0**	**10.0**
yomomma	34	0	0.4	–	**1.0**	**1.0**
zenon	20	0	0	**3.9**	0	0
zork1	102	9.9	32.6	34	38.3	**38.8**
zork3	3[a]	0	0.5	0.1	**3.63**	2.83
ztuu	100	4.9	21.6	9.2	**85.4**[b]	79.1
Winning percentage / counts		11%/3	18%/5	14%/4	**57%/16**	43%/12
					71%/20	

Table 1: Performance on Jericho benchmark games. The best performing agent score per game is **in bold**. The *Winning percentage / counts* row computes the percentage of games the corresponding agent is best. The scores of baselines are from their papers. The games *905*, *anchor*, *awaken*, *deephome* and *moonlit* are omitted because all the methods achieved the same initial scores.
[a] *Zork3* walkthrough does not maximize the score in the first 100 steps but explores more. [b] Our agent discovers some unbounded reward loops in the game *Ztuu*.

the current observation, and it is computationally expensive and unnecessary to retrieve all of such observations. The utility of past observations associated with each object is often time-sensitive in that new observations may entirely or partially invalidate old observations. We thus propose a time-sensitive strategy for retrieving past observations. Specifically, given the detected objects from the current observation, we retrieve the most recent K observations with at least one shared object. The K retrieved observations are sorted by time steps and concatenated to the current observation. The observations from different time steps are separated by a special token. Our RC-based action prediction model treated the concatenated observations as the observation inputs, and no other parts are changed. We use the notation o_t to represent the current observation and the extended current observation interchangeably.

Training Loss We apply the Deep Q-Network (DQN) [11] to update the parameters $\boldsymbol{\theta}$ of our RC-based action prediction model. The loss function is:

$$\mathcal{L}(\theta) = \mathbf{E}_{(o_t,a_t,r_t,o_{t+1})\sim\mathcal{D}}\Big[||Q(o_t,a_t;\theta) - (r_t + \gamma \max_b Q(o_{t+1},b;\theta^-))||\Big]$$

where $\mathcal{D}$ is the experience replay consisting of recent gameplay transition records. Previous work samples transition tuples with immediate positive rewards more frequently to speed up learning [12, 7]. We extend the strategy from transition level to trajectory level. We prioritize transitions from trajectories that outperform the exponential moving average score of recent trajectories.

4 Experiments

We evaluate our proposed methods on the suite of Jericho supported games. We compared to all previous baselines that include recent methods for large action space and partial observation.

***Jericho* Handicaps and Configuration.** The handicaps used by our methods are the same as other baselines. First, we use the Jericho API to check if an action is valid with game-specific templates. All valid templates are considered in action value prediction. Second, we augmented the observation with the textual feedback returned by the command [*inventory*] and [*look*]. Previous work also included the last action or game score as additional inputs. Our model discarded these two types of inputs as we did not observe a significant difference by our model. The maximum game step number is set to 100 following baselines.

Implementation Details. We apply spaCy[3] to tokenize the observations and detect the objects in the observations. We use the 100-dimensional GloVe embeddings as fixed word embeddings. The out-of-vocabulary words are mapped to a randomly initialized embedding. The dimension of Bi-GRU hidden states is 128. The history retrieval window K is 2. For DQN configuration, we use the ϵ-greedy strategy for exploration, annealing ϵ from 1.0 to 0.05. γ is 0.98. We use Adam to update the weights with 10^{-4} learning rate. Other parameters are set to their default values.

Baselines. We compare with all the public results on the Jericho suite, namely TDQN [7], DRRN [9], and KG-A2C [2]. As discussed, our approaches differ from them mainly in the strategies of handling the large action space and partial observability of IF games.

Overall Performance We summarize the performance of our Multi-Paragraph Reading Comprehension DQN (MPRC-DQN) agent and baselines in Table 1. Of the 28 IF games, our MPRC-DQN achieved or improved the state of the art performance on 16 games. The best performing baseline (DRRN) achieved the state-of-the-art performance on only five games.

We include the performance of an RC-DQN agent, which implements our RC-based action prediction model but only takes the current observations as inputs. It also outperformed the baselines by a large margin. After we consider the RC-DQN agent, our MPRC-DQN still has the highest winning percentage, indicating that our RC-based action prediction model has a significant impact on the performance improvement of our MPRC-DQN and the improvement from the multi-passage retrieval is also unneglectable. Moreover, compared to RC-DQN, our MPRC-DQN has another essential advantage of fast convergence despite whether it improves the final scores of games. Finally, our approaches, overall, achieve the new state-of-the-art on 20 games, giving a significant improvement in the field of IF game playing.

5 Conclusion

We formulate the general IF game playing as MPRC tasks, enabling an MPRC-style solution to efficiently address the key IF game challenges on the huge combinatorial action space and the partial observability in a unified framework. Our approaches achieved significant improvement over the previous state-of-the-art on both game scores and training data efficiency. Our formulation also bridges broader NLU/RC techniques to address other critical challenges in IF games for future work, e.g., common-sense reasoning, novelty-driven exploration, and multi-hop inference.

Acknowledgments

We would like to thank Matthew Hausknecht for helpful discussions on the Jericho environments.

[3] `https://spacy.io`

References

[1] A. Adhikari, X. Yuan, M.-A. Côté, M. Zelinka, M.-A. Rondeau, R. Laroche, P. Poupart, J. Tang, A. Trischler, and W. L. Hamilton. Learning dynamic knowledge graphs to generalize on text-based games. *arXiv preprint arXiv:2002.09127*, 2020.

[2] P. Ammanabrolu and M. Hausknecht. Graph constrained reinforcement learning for natural language action spaces. *arXiv*, pages arXiv–2001, 2020.

[3] P. Ammanabrolu and M. Riedl. 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, Volume 1 (Long and Short Papers)*, pages 3557–3565, 2019.

[4] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[5] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[6] M.-A. Côté, Á. Kádár, X. Yuan, B. Kybartas, T. Barnes, E. Fine, J. Moore, M. Hausknecht, L. El Asri, M. Adada, et al. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, pages 41–75. Springer, 2018.

[7] M. Hausknecht, P. Ammanabrolu, M.-A. Côté, and X. Yuan. Interactive fiction games: A colossal adventure. *arXiv preprint arXiv:1909.05398*, 2019.

[8] M. Hausknecht, R. Loynd, G. Yang, A. Swaminathan, and J. D. Williams. Nail: A general interactive fiction agent. *arXiv preprint arXiv:1902.04259*, 2019.

[9] J. He, J. Chen, X. He, J. Gao, L. Li, L. Deng, and M. Ostendorf. 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, 2016.

[10] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[12] K. Narasimhan, T. Kulkarni, and R. Barzilay. 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, 2015.

[13] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[14] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. 2016.

[15] T. Zahavy, M. Haroush, N. Merlis, D. J. Mankowitz, and S. Mannor. Learn what not to learn: Action elimination with deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3562–3573, 2018.

A RC Model for Template Action Details

Observation and *verb* Representation. We tokenize the observation and the $verb$ phrase into words, then embed these words using pre-trained GloVe embeddings [13]. An shared encoder block that consists of LayerNorm [4] and Bidirectional GRU [5] processes the observation and *verb* word embeddings to obtain the separate observation and *verb* representation.

Observation-*verb* Interaction Layers. Given the separate observation and *verb* representation, we apply two attention mechanisms to compute a *verb*-contextualized observation representation. We first apply BiDAF with observation as the context input and $verb$ as the query input. Specifically, we denote the processed embeddings for observation word i and template word j as $\boldsymbol{o}_i$ and $\boldsymbol{t}_j$. The attention between the two words is then $a_{ij}{=}\boldsymbol{w_1}\cdot\boldsymbol{o_i}+\boldsymbol{w_2}\cdot\boldsymbol{t_j}+\boldsymbol{w_3}\cdot(\boldsymbol{o_i}\otimes\boldsymbol{t_j})$, where $\boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{w_3}$ are learnable vectors and $\otimes$ is element-wise product. We then compute the "*verb*2observation" attention vector for the i-th observation word as $\boldsymbol{c_i}{=}\sum_j p_{ij}\boldsymbol{t_j}$ with $p_{ij}{=}\exp(a_{ij})/\sum_j \exp(a_{ij})$. Similarly, we compute the "observation2*verb*" attention vector as $\boldsymbol{q}{=}\sum_i p_i\boldsymbol{o_i}$ with $p_i = \exp(\max_j a_{ij})/\sum_i \exp(\max_j a_{ij})$. We concatenate and project the output vectors as $\boldsymbol{w_4}\cdot\left[\boldsymbol{o_i}, \boldsymbol{c_i}, \boldsymbol{o_i}\otimes\boldsymbol{c_i}, \boldsymbol{q}\otimes\boldsymbol{c_i}\right]$, followed by a linear layer with leaky ReLU activation units [10]. The output vectors are processed by an encoder block. We then apply a residual self-attention on the outputs of the encoder block. The self-attention is the same as BiDAF, but only between the observation and itself.

Observation-Action Value Prediction. We generate an action by replacing the placeholders (arg_0 and arg_1) in a template with objects appearing in the observation. The observation-action value $Q(o, a{=}\langle verb, arg_0{=}obj_m, arg_1{=}obj_n\rangle; \theta)$ is achieved by processing each object's corresponding *verb*-aware observation representation. Specifically, we get the indices of an *obj* in the observation texts $I(obj, o)$. When the object is a noun phrase, we take the index of its headword. Some templates may take zero or one object. We denote the unrequired objects as `none` so that all templates take two objects. The index of the *none* object is for a special token. We set to the index of split token of the observation contents. Because the same object has different meanings when it replaces different placeholders, we apply two GRU-based embedding functions for the two placeholders, to get the object's *verb*-placeholder dependent embeddings. We derive a single vector representation $\boldsymbol{h}_{arg_0=obj_m}$ for the case that the placeholder arg_0 is replaced by obj_m by mean-pooling over the *verb*-placeholder dependent embeddings indexed by $I(obj_m, o)$ for the corresponding placeholder arg_0. We apply a linear transformation on the concatenated embeddings of the two placeholders to obtain the observation action value $Q(o, a){=}\boldsymbol{w}_5\cdot\left[\boldsymbol{h}_{arg_0=obj_m}, \boldsymbol{h}_{arg_1=obj_n}\right]$ for $a{=}\langle verb, arg_0{=}obj_m, arg_1{=}obj_n\rangle$. Our formulation avoids the repeated computation overhead among different actions with a shared template verb phrase.