# Towards Knowledge-Graph Constrained Generation for Text Adventure Games

**Anonymous ACL submission**


## Abstract

Text adventure games provide players with a world to explore the opportunity to interact with it free from the constraints of typical games by using natural language to specify their actions. While revolutionary at the time, these games ultimately came with strict limits on the kinds of inputs they could successfully parse. Recent efforts have made use of large language models to create text adventure games capable of handling any input the user throws at it, but at the cost of lacking any sense of structure or permanence. Is it possible to combine the best aspects of both of these styles of text adventure games, resulting in a game that accepts broad user inputs while remaining grounded? We present one potential method for tackling this problem and discuss both the major difficulties and some potential avenues to explore.


## 1 Introduction

Games like Infocom's *Zork* present players with interactive worlds and stories entirely through the medium of text. These games allow players to describe what they want to accomplish in natural language, expanding their options beyond a small number of enumerated actions (a fact which has also made them an active area of research in reinforcement learning). However, while the number of technically valid actions in a given state is massive, limits on the fidelity of semantic parsers and the contents of game states mean that actions accepted by the game typically amount to a relatively limited set of common verbs composed with objects in the current scene. Far from the promise of being able to do anything they want, players often find their more creative (or outlandish) actions rejected by the game's parser. Recently, the team behind *AI Dungeon* has attempted to solve this problem by leveraging OpenAI's GPT-3 language model. The game can respond to free-form text as input and condition the language model to generate its output,

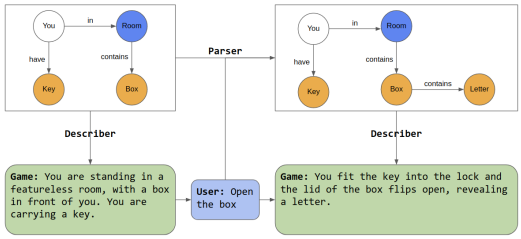


Figure 1: An example of the `Parser` and `Describer` in action.

eliminating the need for a dedicated parser. While effective at accommodating arbitrary user input, the resulting game has no system that enforces the coherence of the world implied by the text – facts about the world established by one output might be contradicted by the next.

Combining these two modes of text adventure games – one limited but grounded, the other free-form but unstructured – is an ambitious goal, but could provide players with a completely novel game-playing experience. While there are many ways this problem could be tackled, one promising method is to leverage knowledge graphs (KGs) as a way to represent the game's internal state. In this conception, a knowledge graph would act as the stable intermediary between the player's inputs and the game's outputs. This idealized game would function as follows:

1. The player specifies an action in natural language.

2. A `Parser` system determines the changes in the knowledge graph caused by the action (for instance, opening a treasure chest might add an edge like [gold, in, chest] to the graph).

3. A `Describer` system converts the knowledge graph updates into a natural language description (for instance, mapping the above update to "Within the chest you find a glittering pile of gold."), allowing the player to specify

another action. Similarly, a `look` command would cause the `Describer` to produce a description of the knowledge sub-graph currently visible to the player.

Something like the `Parser` system has already been developed for the purpose of game-playing (Ammanabrolu and Riedl, 2021a), and so we will focus on the `Describer`: the system responsible for converting the knowledge graph / knowledge graph update into a form the player can interact with. The job of the `Describer` has also been discussed in prior work on generating text adventure games from existing fiction stories (Ammanabrolu et al., 2020). There are, however, a few ways in which this proposed game differs from description generation systems studied in prior work. First, the `Describer` is responsible for more than just generating the flavor text of observable objects or NPCs – it also needs to linguistically render the effects of player actions. Second, the `Describer` must operate without access to a source of ground truth textual descriptions (e.g. a reference short story). Instead, the `Describer` is likely to be data-driven, leveraging prior training in order to generate a plausible description for any arbitrary knowledge graph (while it might be possible to construct a rules-based `Describer`, it is very unlikely to be flexible enough to cover the wide range of inputs the game is designed to accommodate). Broadly speaking, in order to be successful, such a `Describer` system should exhibit three properties:

**Accuracy**: when converting a knowledge graph to natural language, it is crucial that the `Describer` does not fail to convey any of the relations observable to the player or imply the existence of any relations not actually instantiated in the knowledge graph. Each description is the player's only window into the game world, and errors in this representation will interfere with the player's ability to productively interact with the world.

**Consistency**: in order to maintain the structure of games like *Zork*, the `Describer` should ensure that the description of a specific knowledge subgraph does not change too much merely as a function of time: when a player returns to a stable game state after some time away, the description of that state should remain unchanged. Similarly, stylistic elements of the `Describer` should not change too much over time in order to maintain a coherent theme.

**Novelty**: the best `Describer` system is one that incorporates some of the literary language often employed by text adventure games, doing more than merely reciting to the player a list of facts and observed objects. While stylistic liberties should not get in the way of accuracy or consistency, a certain amount of "charm" is important for a game featuring the `Describer` to be worth playing.

## 2 Challenges

### 2.1 Data

Training the `Describer` system requires a large source of data in the form of knowledge graphs and corresponding natural language descriptions. The two most likely current sources of this data are **TextWorld** (Côté et al., 2018) and **Jericho** (Hausknecht et al., 2019). Both systems have proved invaluable for text adventure game research, especially for game-playing, but ultimately neither proves entirely appropriate for training the `Describer`. While TextWorld is capable of procedurally generating text games in a variety of settings, interactions between the player and the world are still largely restricted to small set of repeated behaviors. TextWorld also features a relatively small vocabulary and uses consistent phrasing when describing states. Thus, while it is possible to use this data to train the `Describer`, the resulting system is likely to have difficulty generalizing to unseen KG updates and producing novel, interesting descriptions.

In this regard, the **JerichoWorld** dataset (Ammanabrolu and Riedl, 2021b), which includes oracle-level play traces and knowledge graphs for a variety of real text adventure games, seems more promising (i.e. more likely to include interesting and varied KG updates and state descriptions). However, in order to ensure accuracy for the purpose of training game-playing agents, the JerichoWorld dataset draws its knowledge graphs from each game's internal representation, determining the presence of knowledge graph edges based on whether the objects are explicitly referenced in the game's code. While this guarantees that the most contextually-relevant edges are captured, it unfortunately omits many relations that a human reader might infer from the state description. For example,

from a state description including the passage *"In one corner of the house there is a small window which is slightly ajar"* the knowledge graph makes no mention at all of the *window*. This is likely insufficient for the `Describer`'s task of recovering a description from the knowledge graph.

The alternative approach, then, is to extract the knowledge graph directly from the text of the Jericho dataset using something like Stanford's OpenIE system, as is done in some prior work (Ammanabrolu and Hausknecht, 2020). While promising, existing information extraction (IE) methods do not perform well on text adventure games, often either producing large amounts of mostly redundant edges or failing to annotate seemingly obvious edges. Designing an IE system that works well on text adventure games might first require using human annotators to construct a dataset of complete knowledge graphs derived from game states.

It may also be worth exploring additional sources of data that are not as readily obvious in their use for training the `Describer`. For instance, the ClubFloyd dataset used to train models like CALM (Yao et al., 2020), consists of play traces for a huge number of text adventure games beyond those included in the Jericho dataset. While these play traces do not come with any of the additional helpful features Jericho provides (i.e. direct access to the game state), they could still form a valuable trove of data if accurate knowledge graphs could be extracted from them.

Even further afield, we might consider records of Table-Top Role-Playing Games (TTRPGs) as a source of data. In many ways, text adventure games are designed to emulate the experience of playing a TTRPG, with the computer taking the role of the Game Master (GM). While interactions between players and GMs in TTRPGs do not fall neatly into the *proposed action* followed by *console output* scheme used in text adventure games, it might nonetheless still be possible to filter portions of TTRPG play traces into such a form (i.e. to identify instances in which a player declares an action for their character, perhaps with an accompanying dice roll, and the GM then responds with the outcome of their action). As with the ClubFloyd dataset, a powerful information extraction system would be necessary in order to obtain the paired knowledge graphs that correspond with the textual descriptions.

### 2.2 Evaluation

The second main challenge in training the `Describer` is determining a way to effectively evaluate the model's outputs. Measuring the quality of a proposed knowledge graph description is somewhat like measuring the quality of neural machine translation and inherits many of its difficulties. The most straightforward method would be to directly compare the model output to the reference description, perhaps with fuzzy string matching, and counting a generated descriptions as successful when its overlap with the reference is sufficiently high. However, in many cases it would be possible for the model to produce a description that is semantically equivalent to the reference but phrased in a dramatically different way. An effective evaluation metric should account for this possibility.

A potential alternative to the direct comparison evaluation metric, then, is the BLEU score (or similar translation scoring metric) computed between the generated description and the reference description (Papineni et al., 2002). While this provides an acceptable measure of the generated description's fluency, it too does not give a strong measure of semantic equivalence: even small differences in wording can represent substantially distinct knowledge graph states or updates (consider "The chest is full of gold" and "The chest is *not* full of gold").

Semantic search methods like sentence transformers (Reimers and Gurevych, 2019) seem to have more promise in this regard, as they are designed to produce vectors that usefully represent the aggregate meaning of a piece of text. Thus, it's possible to design an evaluation metric that measures semantic equivalence using the cosine similarity between the embedding of the generated description and the embedding of the reference. The issue with this approach is that many slight changes in wording (e.g. "The chest is full of gold" vs. "The chest is full of gems") will result in very similar sentence embeddings (e.g. due to the semantic similarity of "gold" and "gems") despite the fact that the sentences represent entirely different knowledge graph states. In many text adventure games, subtle distinctions between semantically similar objects prove vital for solving puzzles or general progression, which highlights the importance of capturing these differences in any proposed evaluation scheme.

Perhaps the most promising evaluation metric currently available is to make use of existing Natu-

ral Language Inference (NLI) models. The NLI task, often used as a pre-training step for sentence embedding models, is to determine whether a source sentence logically entails, contradicts, or has no logical relationship to a target sentence. If two sentences logically entail each other, then their semantic content must be equivalent. State-of-the-art NLI systems achieve impressive accuracy levels on this task, and so could plausibly be leveraged to evaluate semantic equivalence between generated and reference descriptions by checking for mutual entailment. This definition of semantic equivalence is used in prior work to help generate "adversarial paraphrases": passages that preserve meaning while being as stylistically different as possible from the reference (Nighojkar and Licato, 2021). Work in adversarial paraphrase generation therefore might be used to help train the `Describer` to generate descriptions that don't merely repeat stylistic patterns observed during training. The primary issue with this evaluation metric is that most NLI systems are designed to operate on individual sentences or short passages. Descriptions of game states have the potential to be quite long (particularly if the scene includes many interactable objects), which might reduce the accuracy of systems checking for mutual entailment.

Finally, a substantial improvement in IE for text games could be leveraged to form an efficient evaluation scheme: the IE system could be used to extract the knowledge graph from the model output (which should not depend on wording) and then the knowledge graph could be compared to the reference using a graph similarity measure. While measuring graph similarity is far from a solved problem, it nonetheless seems promising to use knowledge graphs to partially sidestep the complications inherent in comparing pieces of text.

## 3 Next Steps

There are many exciting directions for future research, ranging from improved information extraction systems, to new sources of data, to novel model architectures for knowledge graph description generation. Of these, enhancements to existing IE systems seem to be particularly worth exploring. A model capable of extracting more accurate and complete knowledge graphs would be beneficial both for leveraging new datasets and for measuring the effectiveness of `Describer` models – in addition to providing utility for research into knowledge graph assisted *play* of text adventure games.

Ultimately, if the issues outlined above can be solved, there seems to be great potential in the future for a text adventure game that leverages knowledge graphs to combine the persistent and grounded structure of classic text adventure games with the great advances in accommodating player creativity offered by more recent games and models.

## References


Prithviraj Ammanabrolu, Wesley Cheung, Dan Tu, William Broniec, and Mark Riedl. 2020. Bringing stories alive: Generating interactive fiction worlds. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, pages 3–9.

Prithviraj Ammanabrolu and Matthew Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces. In *International Conference on Learning Representations*.

Prithviraj Ammanabrolu and Mark Riedl. 2021a. Learning knowledge graph-based world models of textual environments. In *Thirty-fifth Conference on Neural Information Processing Systems*.

Prithviraj Ammanabrolu and Mark Riedl. 2021b. Modeling worlds in text. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. *CoRR*, abs/1806.11532.

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

Animesh Nighojkar and John Licato. 2021. Improving paraphrase detection with the adversarial paraphrasing task. *arXiv preprint arXiv:2106.07691*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. Keep calm and explore: Language models for action generation in text-based games. In *Empirical Methods in Natural Language Processing (EMNLP)*.