TextWorldExpress: Simulating Text Games at One Million Steps Per Second

Peter Jansen
University of Arizona, Tucson, USA
pajansen@arizona.edu

Marc-Alexandre Côté
Microsoft Research Montréal
macote@microsoft.com

Abstract

Text-based games offer a challenging test bed to evaluate virtual agents at language understanding, multi-step problem-solving, and common-sense reasoning. However, speed is a major limitation of current text-based games, capping at 300 steps per second, mainly due to the use of legacy tooling. In this work we present TextWorldExpress, a high-performance simulator that includes implementations of three common text game benchmarks that increases simulation throughput by approximately **three orders of magnitude**, reaching over one million steps per second on common desktop hardware. This significantly reduces experiment runtime, enabling billion-step-scale experiments in about one day.[1] [2] [3]

1 Introduction

One of the long standing goals of artificial intelligence is to create agents that can work and reason in embodied environments. Toward this goal, a variety of virtual environments have been created that allow simulated robots the opportunity to learn to a variety of tasks, in settings from household environments (Kolve et al., 2017; Shridhar et al., 2020a) to Minecraft (Guss et al., 2019). Because high-fidelity 3D virtual environments are challenging and resource intensive to develop, simpler 2D environments have also been proposed (e.g. Chevalier-Boisvert et al., 2019; Küttler et al., 2020) that allow agents to focus on learning skills such as search or navigation in graphically simpler environments.

Recently, text games – or environments rendered entirely in natural language – have emerged as an alternate research methodology for embodied agent research, centrally due to their low barrier to entry compared to 3D games, coupled with their ability to easily model complex tasks at a high-level (see Jansen, 2022, for review). For example, a cooking game might require an agent to read a recipe, find ingredients, then prepare those ingredients to create a meal. Text games model an agent as they navigate an environment, rendering their observations in text (e.g. *"You are in the kitchen. You see..."*). Similarly, agents interact with the environment through abstracted high-level natural language commands (e.g. *"move south"*, or *"pick up carrot"*), rather than lower-level actions common in 3D environments (e.g. *rotate agent 2 degrees clockwise*).

Text games require a variety of common-sense knowledge to complete successfully (Ryu et al., 2022; Murugesan et al., 2021b), including understanding common procedures (such as how to read and follow instructions), as well as affordances about the world – for example, that buildings have rooms, containers must be opened before their con-

Environment Simulator	**SPS**
2D/3D Simulators[4]	
AI2Thor (Kolve et al., 2017)	30†
MineRL (Guss et al., 2019)	180†
BabyAI (Chevalier-Boisvert et al., 2019)	3k
NetHack (Küttler et al., 2020)	14k
MegaVerse (Petrenko et al., 2021)	327k†
Text Game Simulators[5]	
TextWorld (Côté et al., 2018)	300
Jericho (Hausknecht et al., 2020)	1
ScienceWorld (Wang et al., 2022)	20
TextWorldExpress *(online,* Python*)*	32k
TextWorldExpress *(precrawled,* Python*)*	316k
TextWorldExpress *(online,* Java*)*	212k
TextWorldExpress *(precrawled,* Java*)*	4M

Table 1: Single-thread simulation speed of common 2D, 3D, and text-game environment simulators. Speed is measured in terms of Steps Per Second (SPS). † symbolizes that simulation is carried out on GPUs. TextWorldExpress outperforms other text game simulators by approximately three orders of magnitude.

[1]Code: github.com/cognitiveailab/TextWorldExpress
[2]Video: youtu.be/HLFAnRKuTlE
[3]Demo: marccote-textworldexpress.hf.space
[4]Performance reported from (Zholus et al., 2022).
[5]Benchmark scripts provided in the code repository.

tents can be observed or removed, and so forth. As such, text games are still extremely challenging for agents, with current state-of-the-art performance at only 12% for classic interactive fiction games such as *Zork* (Yao et al., 2021; Ammanabrolu et al., 2021). Similarly, interactivity and explicit step-by-step reasoning appears challenging for agents. For example, there appears to be a large dissociation between a model's ability to answer questions about topics (e.g., science exam questions) and its ability to perform very similar experiments in interactive text environments, even with substantial training (Wang et al., 2022). This suggests that explicit interactive multi-step reasoning is still very challenging for contemporary methods like language models, and that accurate procedural knowledge is currently difficult to generate. Together, these highlight the importance of using text games as a vehicle for explicit, embodied, step-by-step reasoning about the world.

To help support these efforts, a number of simulators have recently been developed for text game research, shown in Table 1. Current tooling for text games is built on legacy code bases, providing strong limitations in rendering speed – at present, most simulators are limited to running at between 1 and 300 steps per second. This generally limits agents from using modeling paradigms with fast iteration cycles and high sample requirements (such as reinforcement learning, or evolutionary learning), and restricts users to modeling techniques with large train and inference cycles (such as language models) where the simulator no longer becomes the bottleneck in experiment runtimes.

In this work, we develop a high-speed framework for text-based games in natural language processing research. Our contributions are:

1. TEXTWORLDEXPRESS, a highly optimized simulator that includes reimplementations of three text game benchmarks focusing on instruction following, commonsense reasoning, and object identification, as well as other newer benchmarks for evaluating arithmetic, navigation, and neurosymbolic reasoning.

2. We empirically demonstrate that this simulator runs up to *three orders of magnitude* faster than current tooling, reaching 300k steps per second (SPS) on a single-thread, and exceeding 1M SPS on modest multi-core desktop hardware. This substantially reduces experiment times (from weeks to hours) for sample-heavy machine learning agents.

2 Related Work

Research Paradigm: Text games are a rapidly expanding research paradigm for learning and evaluating situated natural language processing agents on a variety of tasks, with over 100 papers written using this paradigm in the last few years (see Jansen, 2022, for review). This may be in part due to language providing useful abstractions for more efficient exploration and planning (Karch et al., 2020; Colas et al., 2020; Mu et al., 2022; Tam et al., 2022), making task modeling at the level of language more easily approached than with lower-level 3D simulations.

Agent Modeling: Agent modeling has explored a variety of modeling paradigms, including reinforcement learning approaches (Osborne et al., 2021; Xu et al., 2021), combined with reading comprehension techniques (Narasimhan et al., 2015; Tamari et al., 2019; Guo et al., 2020; Yao et al., 2020, 2021), commonsense reasoning (Ryu et al., 2022; Murugesan et al., 2021b), graph-based networks (Ammanabrolu and Riedl, 2019), and neurosymbolic logic (Kimura et al., 2021b; Chaudhury et al., 2021; Kimura et al., 2021a). Most recent agents make use of large pretrained language models (e.g. Devlin et al., 2019), though these can pose challenges both in inference speed, as well as generalization to interactive environments. For example, a model that can correctly answer 90% of multiple choice elementary science exam questions fails to solve text games that test that same knowledge but in a step-by-step procedural setting, even with significant training (Wang et al., 2022).

Simulation Speed: A variety of simulators currently exist for text games, typically focusing on providing domain-general tooling for creating small procedurally generated research environments (e.g. Côté et al., 2018), or interfacing to the existing body of large interactive fiction games such as Zork (Lebling et al., 1979) from the 1980s and 1990s by providing tooling and APIs (Hausknecht et al., 2020). Nearly all frameworks ultimately generate and run games as *Z-machine* code (e.g. Nelson, 2014), an almost 40-year-old domain specific language designed for portability rather than simulation speed. One of the central challenges in building fast research tooling is *valid* *action generation*. Because games implement different sets of actions, and at different levels of granularity, nearly all contemporary agents require the simulator to supply a list of possible valid actions (such as *put coat in closet*) that could be undertaken by the agent at a given time step. Action spaces can be large – hundreds of thousands of action-object combinations are frequently possible at a given step in most games – and existing frameworks (e.g. Hausknecht et al., 2020) built on legacy tooling perform valid action generation by enumerating then running all possible action-object combinations at each game step then recording which ones are valid. This is extremely costly, substantially reducing simulation performance (as shown in Table 1). In this work, TEXTWORLDEXPRESS has been built from the ground-up using heavily optimized and profiled code to quickly render environments while simultaneously generating an exhaustive list of possible next valid actions for agents, greatly speeding simulation time.

3 Environments

TEXTWORLDEXPRESS offers high-speed versions of three popular benchmark environments frequently used in text game research, as well as a number of newer environments for evaluating specific reasoning competencies:

CookingWorld: The CookingWorld environment (Côté et al., 2018) tasks agents with preparing meals by following the instructions in a recipe that is provided in the environment.[6] Agents must first collect required food ingredients (e.g. milk, bell pepper, flour, salt) that can be found in the environment in canonical locations (e.g. kitchen, pantry, supermarket, garden) and containers (e.g. fridge, cupboard). Randomly generated recipes require agents to first use a knife to prepare food by *slicing, dicing,* or *chopping* a subset of ingredients, then additionally using an appropriate heating appliance to *fry, roast,* or *barbeque* the ingredients. If all ingredients are prepared according to the recipe, the agent can use an action to *prepare the meal*, and finally *eat the meal* to complete the task successfully. Task complexity can be controlled by varying the number of locations in the environment, the number of ingredients required for the recipe, and the number of distractor ingredients randomly placed

[6]This task was used as part of the First TextWorld Problems competition (`https://aka.ms/ftwp`) and named by (Madotto et al., 2020).

CookingWorld Game
North of House You are in the kitchen. In one part of the room you see a fridge that is closed. There is also a dining chair, that has nothing on it. You also see a dishwasher that is closed. In another part of the room you see a trash can that is closed. In one part of the room you see an oven. There is also a cutlery drawer that is closed. You also see a stove. In another part of the room you see a counter that has a **cookbook**, a green apple, and a red hot pepper on it. In one part of the room you see a kitchen cupboard that is closed. To the South you see the backyard. To the West you see the pantry. > *read* ***cookbook***
Gather all following ingredients and follow the directions to prepare this tasty meal. Ingredients: **purple potato**, **yellow bell pepper**. Directions: dice the **purple potato**, fry the **purple potato**, roast the **yellow bell pepper**, prepare meal. > *open cutlery drawer*
You open the cutlery drawer. The cutlery drawer contains a **knife**. > *take* ***knife***
You take the knife. > *move south*
You are in the backyard. In one part of the room you see a barbeque. There is also a patio chair, that has nothing on it. You also see a clothes line, that has nothing on it. In another part of the room you see a garden that has a raw **purple potato**, and a **yellow bell pepper** on it. In one part of the room you see a workbench, that has nothing on it. There is also a patio table, that has nothing on it. To the North you see the kitchen. > *take* ***purple potato***
You take the **purple potato**. > *take* ***yellow bell pepper***
You take the **yellow bell pepper**. > *move north*
You are in the kitchen. > *dice* ***purple potato***
You dice the **purple potato**. > *cook* ***purple potato*** *in stove*
You fry the **purple potato** with the stove. > *cook* ***yellow bell pepper*** *with the oven*
You roast the **yellow bell pepper** with the oven. > *prepare meal*
The meal has been added to your inventory. > *eat meal*
Game completed.

Table 2: An example CookingWorld text game, requiring the agent to read a recipe, collect ingredients, and follow preparation instructions. User or agent-entered *actions* for the 12-step solution are italicized. Relevant **task objects** are bolded, while distractor objects are underlined. Here, an easy-level parametric game was generated consisting of 3 locations *(kitchen, backyard, pantry)*, 2 required ingredients *(potato, bell pepper)*, and 2 distractor ingredients *(apple, hot pepper)*.

in the environment that are not required for the recipe. The recipes and environments are parametrically generated, with subsets of ingredients and specific preparations held out between training, development, and test sets to prevent overfitting. An example CookingWorld task is shown in Table 2.

TextWorld Commonsense (TWC): Text game agents frequently learn the dynamics of environment – such as the need to open a door before one can move through it – from interacting with the environment itself, rather than using a preexisting knowledge base of common sense facts or object affordances that would speed task learning. TextWorld Commonsense (Murugesan et al., 2021a) aims to evaluate agents on common sense knowledge that can not be directly learned from the environment by providing agents a clean-up task where the agent must place common household objects (e.g. *a dirty dish*) in their canonical locations (e.g. *the dishwasher*) that can be found in knowledge bases such as ConceptNet (Liu and Singh, 2004; Speer et al., 2017). Separate lists of objects are used in the training, development, and test sets, meaning the agent can not learn object locations from the training set alone, and must rely on an external common sense knowledge base to perform well on the development and test sets. Murugesan et al. (2021a) specify three task difficulty levels, with the easiest including a single location and object to put away, while the hard setting includes two location and up to 7 objects.

Coin Collector: Agents frequently find tasks such as object search, environment navigation, or pick-and-place tasks challenging (Shridhar et al., 2020b). The Coin Collector game (Yuan et al., 2018) distills these into a single benchmark where an agent must explore a series of rooms to locate and pick up a single coin. In the original implementation, the game map typically takes the form of a connected loop or chain, such that continually moving to new locations means the agent will eventually discover the coin – while including medium and hard modes that add in one or more "dead-end" paths. To control for environment difficulty across games, the TEXTWORLDEXPRESS reimplementation uses the same map generator across environments, and generates arbitrary home environments rather than connected loops. The user maintains control of other measures of difficulty, including the total number of rooms, and the number of distractor objects placed in the environment.

Adding new games: New games can be added to TEXTWORLDEXPRESS, and 4 additional games that benchmark arithmetic, navigation, and neurosymbolic reasoning have been added since its initial release[7]. Adding new games takes about a day of coding, which can be more effortful than using the domain-specific implementation languages of existing game engines (e.g. Côté et al., 2018).

[7]See full list at `https://github.com/cognitiveailab/TextWorldExpress#environments`.

Action	Description
Generic actions	
look around	*describe current location*
inventory	*list agent inventory*
examine **OBJ**	*examine an object*
move **DIR**	*move north, east, south, or west*
open **OBJ**	*open a door or container*
close **OBJ**	*close a door or container*
take **OBJ**	*pick up an object*
put **OBJ** in **OBJ**	*put an object in a container*
Extended actions (CookingWorld)	
read **OBJ**	*read a recipe book*
cook **OBJ** in **OBJ**	*cook an ingredient*
chop **OBJ**	*chop an ingredient*
slice **OBJ**	*slice an ingredient*
dice **OBJ**	*dice an ingredient*
eat **OBJ**	*eat an ingredient*
prepare meal	*prepare the meal*

Table 3: The action space of the environments, as well as descriptions of each action. Actions can take zero, one, or two object (OBJ) or direction (DIR) arguments.

3.1 Action Space and Valid Action Generation

The three benchmark games each have up to 15 different types of actions available to agents, described in Table 3. These include common text-game actions such as *taking objects*, *moving locations*, and *opening doors*, as well as domain-specific actions such as *slicing* or *cooking ingredients* for the cooking-domain game. Actions may take zero (e.g. *look around*), one (e.g. *take shirt*), or two (e.g. *put shirt in closet*) objects as arguments.

Most contemporary high-performance game agents (e.g. Ammanabrolu and Hausknecht, 2020; Murugesan et al., 2021a) make use of a "valid-action handicap" – that is, at each step, they require a list of possible valid actions that can be taken in the environment, from which they select a single action to undertake. For example, a kitchen agent might wish to *dice the carrot*, but such an action would only be available to the agent if it currently possessed both a carrot and a knife in its inventory. This valid-action detection is typically implemented overtop of existing interactive fiction games (such as *Zork*) by an interface framework (e.g., Jericho; Hausknecht et al., 2020) at significant loss to the simulation framerate. In contrast, TEXTWORLDEXPRESS was designed from the ground-up to provide fast valid action generation to maintain high framerates.

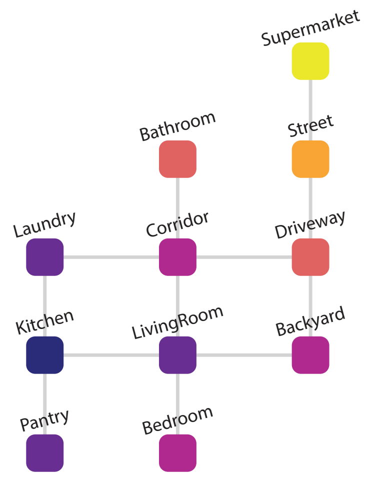

Figure 1: An example of the random map generation process, containing 11 separate locations. Locations are iteratively placed on a 7x7 grid, then interconnected (edges) on the four cardinal directions *(north, east, south, west)* based on connection preferences. For example, the *Pantry* prefers to connect to the *Kitchen*, and will never connect to the *Bedroom*.

3.2 Map Generation

Navigation tasks – such as exploring an environment, or navigating to a specific location – are typically challenging for contemporary text game agents. Because of this, games typically reduce the burden of navigation by providing simplified maps. At one end of the extreme, the original *TextWorld Commonsense* uses small maps containing only one or two locations, while at the other extreme *CookingWorld* creates maps with over a dozen locations interconnected in common ways (i.e. a *kitchen* is usually connected to a *pantry*, *backyard*, and/or *corridor*, but is never directly connected to a *supermarket*). To control for the difficulty of the navigation task across environments, TEXTWORLDEXPRESS uses the same map generator across all three benchmark games, while allowing the user to specify parameters such as the number of map locations to control the difficulty of the navigation task.

Environments can consist of up to 11 locations, consisting of locations common to both the *TWC* and *CookingWorld* games. Maps are randomly generated at the start of each game, and allow navigation on four cardinal directions (north, south, east, west). Optionally, rooms may be connected with doors that an agent is required to open before allowing passage, increasing task complexity. Figure 1 shows an example map produced by the generator.

3.3 Object Library

Task objects, room objects, and distractor objects are populated from the object libraries provided by the TextWorld Commonsense and CookingWorld games. This results in approximately 500 possible objects that can populate environments, including containers (e.g. *fridge*, *shelf*, *countertop*), and movable objects (e.g. *red onion, dirty shirt*).

3.4 Parametric Variation

To reduce overfitting, generated tasks and environments vary in their requirements and presentation. Tasks typically vary in task-critical objects, such as the specific objects that need to be cleaned up in *TextWorld Commonsense*, or the recipe, ingredients, and their locations in *CookingWorld*. Environments parametrically vary, centrally in the environment map (how the rooms are interconnected), while also allowing different numbers of distractor objects to be generated in different randomized locations in the environment. Critically, games are deterministic and the generation is repeatable and controlled by a single random seed, such that the same game can be regenerated during agent training and evaluation. To create independent train, development, and test sets, in addition to each game having specific task objects that are unique across training and evaluation sets, we also assign blocks of random seeds to the train, development, and sets. This allows generating thousands of possible parametric variations for each set, while ensuring that the tasks and environments remain unique.

3.5 Scoring

At each time step, the simulator provides the agent a score that signifies the agent's progress in solving a given task. Games typically assign rewards for critical task steps, such as picking up correct ingredients, or preparing ingredients correctly. Because the total score required to complete a game can vary both across games and across task complexity, scores are provided both as raw counts, as well as normalized to between zero (no task progress) and one (task completion). Each game has specific success and failure criterion, which are automatically detected by the simulator, and provided to the agent by the API. For example, if a recipe requires a *carrot* to be *chopped*, but the agent instead *slices* it, this will cause a task failure, and can be used as a reward signal for the agent model to use in adjusting its action policy.

4 Speed Comparison

4.1 Online and Precrawled Generation

To enable extremely fast simulations, TEXTWORLDEXPRESS supports two game generation modes: normal (online) generation, and precrawled generation. In *online generation*, games are parametrically generated and played at runtime, allowing a large number of parametric game variations to be generated, and games to be played up to any number of steps. Conversely, where speed is of critical importance, the simulator supports *precrawling* all possible paths an agent might take in a given environment, and pre-caching these to disk as a JSON file. This allows extremely fast game playing – at essentially the speed of updating a pointer to a particular node in the precrawled state tree – at the expense of generating and loading large files, that pragmatically limit the total number of steps that can be crawled and precached in the environment.[8] Precrawling is a unique feature offered by TEXTWORLDEXPRESS, as games taking minutes to crawl in this framework can take days or weeks to crawl in TEXTWORLD.

4.2 Evaluating Simulation Speed

We empirically compare the simulation speed of TEXTWORLDEXPRESS with three frameworks.

TextWorld (Côté et al., 2018) is a framework for generating parametric text games for natural language processing research. Games are specified using predicate logic (to define action rules) and a context-free grammar (to generate text), which TextWorld reformulates into Inform7 code (Nelson, 2006), that is then ultimately compiled to a Z-Machine game (Nelson, 2014). The three benchmark games reimplemented in TEXTWORLDEXPRESS were originally implemented in TextWorld.

Jericho (Hausknecht et al., 2020) provides a research interface to the existing body of interactive fiction games, such as Zork (Lebling et al., 1979), that were originally written for the Z-Machine interpreter. Critically, Jericho provides facilities for action template extraction and valid-action generation, to reduce the difficulty of interfacing classic interactive fiction games with language agents.

ScienceWorld (Wang et al., 2022) is a science-domain text game simulator that provides the ability to train and evaluate agents on scientific tasks normally learned by elementary science students, such as changes of states of matter (melting, boiling, freezing), life cycles of plants and animals, and basic chemistry. Supporting this is a series of complex simulation engines (e.g., thermodynamics, electrical conductivity, genetics) which increase simulation fidelity at the cost of speed. Similar to TextWorld and Jericho, ScienceWorld supports generating valid actions at each time step.

The results of this evaluation, using random agents to traverse the environments, are shown in Table 1. The highly optimized TEXTWORLDEXPRESS is able to simulate games in *online* generation mode at an average of *212k* frames per second per thread, or nearly three orders of magnitude faster than other frameworks.[9] This varies between *256k* steps per second for the fastest environment with the least complex action space *(Coin Collector)*, to *155k* steps per second for the most complex action space *(CookingWorld)*. On an 8-core workstation, this enables million-step experiments to be simulated per second, with billion-step experiments possible in approximately one hour.[10] In contrast, one billion steps would take approximately 38 days using the original TextWorld implementations. In *precrawled* mode, where game states are precached, single-thread speeds of up to 4 million steps per second are possible. Our fastest multi-threaded benchmark on desktop hardware (an AMD 3950X 16-core, 32-thread CPU) reaches 34M steps per second, enabling billion-step-scale simulations in approximately 30 seconds.

5 Conclusion

We present TEXTWORLDEXPRESS, a fast simulator for text-game research that reimplements three benchmark environments while running three orders of magnitude faster than their original implementations. New games can be added using existing games as templates, and four new games benchmarking specific reasoning competencies like arithmetic and navigation have been added since its initial release. The simulator supports common features (such as valid action detection), while providing new enabling features, such as quickly precrawling entire game state trees. This work is released as open source.

[8] As an example, a 1GB file can typically store precrawled game trees for a single game variation up to between 8 and 12 steps, depending on the complexity of the action space.

[9] PYTHON performance is 10X slower than JAVA/SCALA performance due to the speed of PYTHON-JVM binders.

[10] Using pre-crawled paths, we managed to run *billion-game* experiment on a 32-core server in about a day.

6 Broader Impacts

Embodied agents require a variety of commonsense reasoning skills and competencies about the world in order to successfully perform tasks. Text games distill task learning to a high level of abstraction, allowing conceptual-level procedural knowledge to be acquired without simultaneously learning challenging low-level perceptual or motor tasks as in 3D simulators (e.g. Shridhar et al., 2020a; Petrenko et al., 2021), while reducing the computational requirements to run experiments from expensive GPU servers to common desktop hardware. Futher, Shirdhar et al. (2020b) have empirically demonstrated that agents can be inexpensively pretrained on tasks in a text world environment, then transfer much of their performance to more realistic 3D environments, speeding training. TEXTWORLDEXPRESS, which increases the speed of text game experiments by three orders of magnitude, enables running experiments faster, at greater scale, or using alternate sample-heavy machine learning frameworks than currently available simulators.

7 Limitations

TEXTWORLDEXPRESS has two main limitations compared to existing simulators. TEXTWORLDEXPRESS gains much of its speed by developing a highly-profiled simulator with hard-coded implementations of text games. Unlike the original TEXTWORLD simulator, which is designed to allow new environments to be implemented with a domain-specific language, adding new environments to TEXTWORLDEXPRESS is more effortful and requires coding in SCALA, a derivative of JAVA. Similarly, for speed, the TEXTWORLDEXPRESS user input parser is simplified, and it only recognizes valid actions as it presents them to the agent, without facilities for alternate surface forms, misspellings, or other variations. While it is common for agents to select actions from a valid action list, the lack of a diverse input parser limits utility for human participants who might choose to play these games.

Acknowledgements

This work supported in part by National Science Foundation (NSF) award #1815948 to PJ, and gift from the Allen Institute for Artificial Intelligence (AI2).

References

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. 2019. 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, Minneapolis, Minnesota. Association for Computational Linguistics.

Prithviraj Ammanabrolu, Jack Urbanek, Margaret Li, Arthur Szlam, Tim Rocktäschel, and Jason Weston. 2021. How to motivate your dragon: Teaching goal-driven agents to speak and act in fantasy worlds. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 807–833.

Subhajit Chaudhury, Prithviraj Sen, Masaki Ono, Daiki Kimura, Michiaki Tatsubori, and Asim Munawar. 2021. Neuro-symbolic approaches for text-based policy learning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3073–3078, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2019. Babyai: First steps towards grounded language learning with a human in the loop. In *International Conference on Learning Representations*, volume 105.

Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter Ford Dominey, and Pierre-Yves Oudeyer. 2020. Language as a cognitive tool to imagine goals in curiosity-driven exploration. *ArXiv*, abs/2002.09253.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben A. Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew J. Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. In *CGW@IJCAI*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7755–7765, Online. Association for Computational Linguistics.

William H Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. 2019. Minerl: a large-scale dataset of minecraft demonstrations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2442–2448.

Matthew J. Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *AAAI*.

Peter A Jansen. 2022. A systematic survey of text worlds as embodied natural language environments. In *WordPlay Workshop: When Language Meets Games*.

Tristan Karch, Nicolas Lair, Cédric Colas, Jean-Michel Dussoux, Clément Moulin-Frier, Peter Ford Dominey, and Pierre-Yves Oudeyer. 2020. Language-goal imagination to foster creative exploration in deep rl.

Daiki Kimura, Subhajit Chaudhury, Masaki Ono, Michiaki Tatsubori, Don Joven Agravante, Asim Munawar, Akifumi Wachi, Ryosuke Kohita, and Alexander Gray. 2021a. LOA: Logical optimal actions for text-based interaction games. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 227–231, Online. Association for Computational Linguistics.

Daiki Kimura, Masaki Ono, Subhajit Chaudhury, Ryosuke Kohita, Akifumi Wachi, Don Joven Agravante, Michiaki Tatsubori, Asim Munawar, and Alexander G. Gray. 2021b. Neuro-symbolic reinforcement learning with first-order logic. In *EMNLP*.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. 2017. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*.

Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. 2020. The nethack learning environment. *Advances in Neural Information Processing Systems*, 33:7671–7684.

P David Lebling, Marc S Blank, and Timothy A Anderson. 1979. Zork: a computerized fantasy simulation game. *Computer*, 12(04):51–59.

Hugo Liu and Push Singh. 2004. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Andrea Madotto, Mahdi Namazifar, Joost Huizinga, Piero Molino, Adrien Ecoffet, Huaixiu Zheng, Alexandros Papangelis, Dian Yu, Chandra Khatri, and Gokhan Tur. 2020. Exploration based language learning for text-based games. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 1488–1494. International Joint Conferences on Artificial Intelligence Organization. Main track.

Jesse Mu, Victor Zhong, Roberta Raileanu, Minqi Jiang, Noah D. Goodman, Tim Rocktaschel, and Edward Grefenstette. 2022. Improving intrinsic exploration with language abstractions. *ArXiv*, abs/2202.08938.

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

Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. 2021b. Efficient text-based reinforcement learning by jointly leveraging state and commonsense graph representations. In *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, Online. Association for Computational Linguistics.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. 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, Lisbon, Portugal. Association for Computational Linguistics.

Graham Nelson. 2006. Natural language, semantic analysis, and interactive fiction. *IF Theory Reader*, 141:99–104.

Graham Nelson. 2014. The z-machine standards document version 1.1.

Philip Osborne, Heido Nomm, and André Freitas. 2021. A survey of text games for reinforcement learning informed by natural language. *ArXiv*, abs/2109.09478.

Aleksei Petrenko, Erik Wijmans, Brennan Shacklett, and Vladlen Koltun. 2021. Megaverse: Simulating embodied agents at one million experiences per second. In *International Conference on Machine Learning*, pages 8556–8566. PMLR.

Dongwon Ryu, Ehsan Shareghi, Meng Fang, Yunqiu Xu, Shirui Pan, and Reza Haf. 2022. Fire burns, sword cuts: Commonsense inductive bias for exploration in text-based games. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 515–522, Dublin, Ireland. Association for Computational Linguistics.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020a. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020b. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*.

Allison C. Tam, Neil C. Rabinowitz, Andrew Kyle Lampinen, Nicholas A. Roy, Stephanie C. Y. Chan, DJ Strouse, Jane X. Wang, Andrea Banino, and Felix Hill. 2022. Semantic exploration from language abstractions and pretrained representations. *ArXiv*, abs/2204.05080.

Ronen Tamari, Hiroyuki Shindo, Dafna Shahaf, and Yuji Matsumoto. 2019. Playing by the book: An interactive game approach for action graph extraction from text. In *Proceedings of the Workshop on Extracting Structured Knowledge from Scientific Publications*, pages 62–71, Minneapolis, Minnesota. Association for Computational Linguistics.

Ruoyao Wang, Peter Alexander Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? *ArXiv*, abs/2203.07540.

Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, and Chengqi Zhang. 2021. Generalization in text-based games via hierarchical reinforcement learning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1343–1353, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Xingdi Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Rémi Tachet des Combes, Matthew J. Hausknecht, and Adam Trischler. 2018. Counting to explore and generalize in text-based games. *ArXiv*, abs/1806.11525.

Artem Zholus, Alexey Skrynnik, Shrestha Mohanty, Zoya Volovikova, Julia Kiseleva, Arthur Szlam, Marc-Alexandre Côté, and Aleksandr I. Panov. 2022. IGLU gridworld: Simple and fast environment for embodied dialog agents. *CoRR*, abs/2206.00142.