A Survey of Text Games for Reinforcement Learning Informed by Natural Language

Philip Osborne
Department of Computer Science
University of Manchester
United Kingdom
philiposbornedata@gmail.com

Heido Nõmm
Department of Computer Science
University of Manchester
United Kingdom
heidonomm@gmail.com

André Freitas
Department of Computer Science
University of Manchester
United Kingdom
andre.freitas@manchester.ac.uk

Abstract

Reinforcement Learning has shown success in a number of complex virtual environments. However, many challenges still exist towards solving problems with natural language as a core component. Interactive Fiction Games (or Text Games) are one such problem type that offer a set of safe, partially observable environments where natural language is required as part of the Reinforcement Learning solution. Therefore, this survey's aim is to assist in the development of new Text Game problem settings and solutions for Reinforcement Learning informed by natural language. Specifically, this survey: 1) introduces the challenges in Text Game Reinforcement Learning problems, 2) outlines the generation tools for rendering Text Games and the subsequent environments generated, and 3) compares the agent architectures currently applied to provide a systematic review of benchmark methodologies and opportunities for future researchers.

1 Introduction

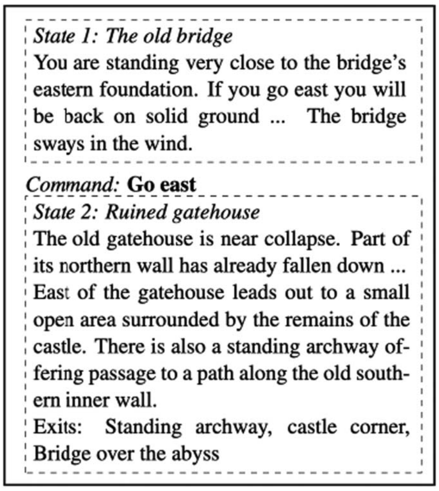
Language is often used by humans to abstract, transfer, and communicate knowledge of their decision making when completing complex tasks. However, traditional Reinforcement Learning (RL) methods, such as the prominent neural agents introduced by Mnih et al. (2013) and Silver et al. (2016), are limited to single task environments defined and solved without any language.

Luketina et al. (2019) specified that further studies are required for improving solutions on problems that necessitate the use of language. The authors also note that language may improve solutions on problems that are traditionally solved without language. This includes the use of neural agents pre-trained on natural language corpora transferring syntactic and semantic information to future tasks. An example is the use of the similarities between the tasks of *chopping* and *cutting* a carrot in CookingWorld (Trischler et al., 2019)—that is, completing one should allow you to transfer the decision making to the other.

To aid in developing solutions on these problems, we pose Text Games as a testing environment as they simulate complex natural language problems in controllable settings. In other words, researchers can generate the Text Games with limitations to evaluate any number of the specific challenges given by Dulac-Arnold et al. (2019) and Luketina et al. (2019). Detailed descriptions of the possible controls (defined as 'handicaps') are provided in Section 2.4. Likewise, the challenges with currently posed solutions are provided in Section 2.3 and are summarized by the following:

- **Partial observability** - observed information is only a limited view of the underlying truth (shown in Figure 1).
- **Large state space** - balancing exploration to new states against exploiting known paths.
- **Large and sparse action space** - language increases the number of actions as there are multiple ways to describe the same input.
- **Long-term credit assignment** - learning which actions are important when reward

Transactions of the Association for Computational Linguistics, vol. 10, pp. 873–887, 2022. https://doi.org/10.1162/tacl_a_00495
Action Editor: Marco Baroni. Submission batch: 12/2021; Revision batch: 3/2022; Published 8/2022.



State 1: The old bridge
You are standing very close to the bridge's eastern foundation. If you go east you will be back on solid ground ... The bridge sways in the wind.

Command: **Go east**

State 2: Ruined gatehouse
The old gatehouse is near collapse. Part of its northern wall has already fallen down ... East of the gatehouse leads out to a small open area surrounded by the remains of the castle. There is also a standing archway offering passage to a path along the old southern inner wall.
Exits: Standing archway, castle corner, Bridge over the abyss

Figure 1: Sample gameplay from a fantasy Text Game as given by Narasimhan et al. (2015) where the player takes the action 'Go East' to cross the bridge.

signals might only be received at the end of many successive steps.

- **Understanding parser feedback and language acquisition** - how additional language may be grounded into the problem itself.
- **Commonsense reasoning** - how to utilize contextual knowledge of the problem to improve the solutions.
- **Knowledge representation** - using graph representation of knowledge for improved planning.

Text Games are also safe in that we are not making decisions on a 'live' system (i.e., one that affects the real-world). This allows agents to explore freely without limitations and is often a requirement of training any Reinforcement Learning agent. Furthermore, this has also been used to evaluate methods with unique goals such as Yuan et al. (2018), who wanted to maximize exploration to new states and achieved this by adding a reward to the first encounter of an unseen state.

So far, research has mostly been performed independently, with many authors generating their own environments to evaluate their proposed architectures. This lack of uniformity in the environments makes comparisons between authors challenging and a need for structuring recent work is essential for systematic comparability.

Formally, this survey provides the first systematic review of the challenges posed by the generation tools designed for Text Game evaluation. Furthermore, we also provide details on the currently generated environments and the resultant RL agent architectures used to solve them. This acts as a complement to prior studies in the challenges of real-world RL and RL informed by natural language by Dulac-Arnold et al. (2019) and Luketina et al. (2019), respectively.

2 Text Games and Reinforcement Learning

Text Games are turn-based games that interpret and execute player commands to update the current position within the game environment. The current state is provided to the player in language and the player must take actions by entering textual commands that are parsed and confirmed to be valid or not. Many of these games were designed by human developers for human players based on real-world logic with a clear definition of how a player may win the game (e.g., Zork [Anderson et al., 1980]). In many cases, common knowledge is used to complete the task efficiently as, for example, a key encountered at one point will be needed to progress through locked doors later in the game.

This creates a complex set of decisions that are connected by contextual knowledge of how objects interact with each other to reach a clear goal. Reinforcement Learning is a natural solution, as the problem can be defined by a sequence of states dependent on the player's actions with a reward signal defined on the game's win condition. However, an agent will also need to find an efficient solution to the linguistic challenges that a human player would experience when exploring through the text-based game.

In this section, we first formalize the Reinforcement Learning model before introducing the Text Game generators, challenges, and possible controls to provide the essential background to the published methods introduced in later sections.

2.1 Environment Model

Reinforcement Learning is a framework that enables agents to reason about sequential decision making problems as an optimization process (Sutton and Barto, 1998). Reinforcement Learning requires a problem to be formulated as a Markov Decision Process (MDP) defined by a tuple $< S, A, T, R, \gamma >$ where S is the set of states, A is the set of actions, T is the transition probability function, R the reward signal, and γ the discount factor.

Given an environment defined by an MDP, the goal of an agent is to determine a policy $\pi(a|s)$ specifying the action to be taken in any state that maximizes the expected discounted cumulative return $\sum_{k=0}^{\infty} \gamma^k r_{k+1}$.

In Text Games however, the environment states are never directly observed, but rather, textual feedback is provided after entering a command. As specified by Côté et al. (2018), a Text Game is a discrete-time Partially Observed Markov Decision Process defined by $< S, A, T, \Omega, O, R, \gamma >$ where we now have the addition of the set of observations (Ω) and a set of conditional observed probabilities O. Specifically, the function O selects from the environment state what information to show to the agent given the command entered to produce each observation $o_t \in \Omega$.

For example, instead of being provided with $[x, y]$ coordinates or the room's name to define the current state, you are instead given a description of your location as *'You an in a room with a toaster, a sink, fridge and oven.'*. One might deduce in this example that the player is likely in a kitchen but it raises an interesting idea that another room may contain one of these objects and not be the kitchen (e.g., fridges/freezers are also often located in utility rooms and sinks are in bathrooms). The challenge of partial observability is expanded on in Section 2.3.

2.2 Text Game Generation

Before we introduce the challenges and handicaps it is important to understand how Text Games are generated as this defines much of the game's complexity in both decision making and the language used. There are two main generation tools for applying agents to interactive fiction games: TextWorld and Jericho.

TextWorld (Côté et al., 2018) is a logic engine to create new game worlds, populating them with objects and generating quests that define the goal states (and subsequent reward signals). It has been used as the generation tool for Treasure Hunter (Côté et al., 2018), Coin Collector (Yuan et al., 2018), CookingWorld (Trischler et al., 2019), and the QAit Dataset (Yuan et al., 2019).

Jericho (Hausknecht et al., 2019a) was more recently developed as a tool for supporting a set of human-made interactive fiction games that cover a range of genres. These include titles such as Zork and Hitchhiker's Guide to the Galaxy. Unsupported games can also be played through Jericho but will not have the point-based scoring system that defines the reward signals. Jericho has been used as the generator tool for CALM (Yao et al., 2020) and Jericho QA (Ammanabrolu et al., 2020a).

Environments built from the TextWorld generative system binds the complexity by the set of objects available. For example, Côté et al. (2018) introduce 10 objects including the logic rules for doors, containers, and keys, where complexity can be increased by introducing more objects and rules into the generation process. The most challenging environments are defined in Jericho, as these contain 57 real Interactive Fiction games that have been designed by humans, for humans. Specifically, these environments contain more complexity in forms of stochasticity, unnatural interactions and unknown objectives—difficulties originally created to trick and hamper players.

More detailed descriptions of the environments generated from these are provided in Section 3.

2.3 Challenges and Posed Solutions

The design and partially observed representation of Text Games creates a set of natural challenges related to Reinforcement Learning. Furthermore, a set of challenges specific to language understanding and noted by both Côté et al. (2018) and Hausknecht et al. (2019a) are given in detail in this section.

Partial Observability The main challenge for agents solving Textual Games is the environment's partial observability; when observed information is only representative of a small part of the underlying truth. The connection between the two is often unknown and can require extensive exploration and failures for an agent to learn from observations and how this relates to its actions.

A related additional challenge is that of *causality*, which is when an agent moves away from a state to the next without completing prerequisites of future states. For example, an agent is required to use a lantern necessary to light its way but may have to backtrack to previous states if this has not been obtained yet; an operation that becomes more complex as the length of the agent's trajectory increases the further into the game they have to backtrack.

Handcrafted reward functions have been proved to work for easier settings, like CoinCollector (Yuan et al., 2018), but more challenging Text Games can require more nuanced approaches. Go-Explore has been used to find high-reward trajectories and discover under-explored states (Madotto et al., 2020; Ammanabrolu et al., 2020a) where more advanced states are given higher priority over states seen earlier on in the game by a weighted random exploration strategy. Ammanabrolu et al. (2020a) have expanded on this with a modular policy approach aimed at noticing bottleneck states with a combination of a patience parameter and intrinsic motivation for new knowledge. The agent would learn a chain of policies and backtrack to previous states upon getting stuck. Heuristic-based approaches have been used by Hausknecht et al. (2019b) to restrict navigational commands to only after all other interactive commands have been exhausted.

Leveraging past information has been proven to improve model performance as it limits the partial observability aspect of the games. Ammanabrolu and Hausknecht (2020) propose using a dynamically learned Knowledge Graph (KG) with a novel graph mask to only fill out templates with entities already in the learned KG.

Large State Space Whereas Textual Games, like all RL problems, require some form of explofocusration to find better solutions, some papers focus specifically on countering the natural overfitting of RL by actively encouraging exploration to unobserved states in new environments. For example, Yuan et al. (2018) achieved this by setting a reward signal with a bonus for encountering a new state for the first time. This removes the agent's capability for high-level contextual knowledge of the environment in favor of simply searching for unseen states.

Subsequent work by Côté et al. (2018)—Treasure Hunter—has expanded on this by increasing the complexity of the environment with additional obstacles such as locked doors that need color matching keys requiring basic object affordance and generalization ability. In a similar vein to Treasure Hunter, where in the worst case agents have to traverse all states to achieve the objective, the *location* and *existence* settings of QAit (Yuan et al., 2019) require the same with addition of stating the location or existence of an object in the generated game. These solutions are also related to the challenge of *Exploration vs Exploitation* that is commonly referenced in all RL literature (Sutton and Barto, 1998).

Large, Combinatorial, and Sparse Action Spaces Without any restrictions on length or semantics, RL agents aiming to solve games in this domain face the problem of an unbounded action space. Early works limited the action phrases to two word sentences for a verb-object, more recently combinatory action spaces are considered that include action phrases with multiple verbs and objects. A commonly used method for handling combinatory action spaces has been to limit the agent to picking a template T and then filling in the blanks with entity extraction (Hausknecht et al., 2019a; Ammanabrolu et al., 2020a; Guo et al., 2020).

Two other approaches have been used: (i) action elimination (Zahavy et al., 2018; Jain et al., 2020), and (ii) generative models (Tao et al., 2018; Yao et al., 2020). The first aims to use Deep Reinforcement Learning with an Action Elimination Network for approximation of the admissibility function: whether the action taken changes the underlying game state or not. The second has been used in limited scope with pointer softmax models generating commands over a fixed vocabulary and the recent textual observation. The CALM generative model, leveraging a fine-tuned GPT-2 for textual games, has proved to be competitive against models using valid action handicaps.

Long-Term Credit Assignment Assigning rewards to actions can be difficult in situations when the reward signals are sparse. Specifically, positive rewards might only be obtained at the successful completion of the game. However, environments where an agent is unlikely to finish the game through random exploration provide rewards for specific subtasks such as in Murugesan et al. (2020a), Trischler et al. (2019), and Hausknecht et al. (2019a). The reward signal structured in this way also aligns with hierarchical approaches such as in Adolphs and Hofmann (2020). Lastly, to overcome the challenges presented with reward-sparsity, various hand-crafted reward signals have been experimented with (Yuan et al., 2018; Ammanabrolu et al., 2020a).

Understanding Parser Feedback and Language Acquisition LIGHT (Urbanek et al., 2019) is a crowdsourced platform for the experimentation of grounded dialogue in fantasy settings. It differs from the previous action-oriented environments by requiring dialogue with humans, embodied agents, and the world itself as part of the quest completion. The authors design LIGHT to investigate how ‘a model can both speak and act grounded in perception of its environment and dialogue from other speakers’.

Ammanabrolu et al. (2020b) extended this by providing a system that incorporates ‘1) large-scale language modelling based commonsense reasoning pre-training to imbue the agent with relevant priors and 2) a factorized action space of commands and dialogue’. Furthermore, evaluation can be performed against a dataset collected of held-out human demonstrations.

Commonsense Reasoning and Affordance Extraction As part of their semantic interpretation, Textual Games require some form of commonsense knowledge to be solved. For example, modeling the association between actions and associated objects (*opening* doors instead of *cutting* them, or the fact that *taking* items allows the agent to use them later on in the game). Various environments have been proposed for testing procedural knowledge in more distinct domains and to assess the agent’s generalization abilities.

For example, Trischler et al. (2019) proposed the ‘First TextWorld Problems’ competition with the intention of setting a challenge requiring more planning and memory than previous benchmarks. To achieve this, the competition featured ‘thousands of unique game instances generated using the TextWorld framework to share the same overarching theme—an agent is hungry in a house and has a goal of cooking a meal from gathered ingredients’. The agents therefore face a task that is more hierarchical in nature as cooking requires abstract instructions that entail a sequence of high-level actions on objects that are solved as sub-problems. Furthermore, TW-Commonsense (Murugesan et al., 2020a) is explicitly built around agents leveraging prior commonsense knowledge for object-affordance and detection of out of place objects.

Two pre-training datasets have been proposed that form the evaluation goal for specialized modules of RL agents. The ClubFloyd dataset[1] provides human playthroughs of 590 different text-based games, allowing to build priors and pre-train generative action generators. Likewise, the Jericho-QA (Ammanabrolu et al., 2020a) dataset provides context at a specific timestep in various classical IF games supported by Jericho, and a list of questions, enabling pre-training of QA systems in the domain of Textual Games. They also used the dataset for fine-tuning a pre-trained LM for building a question-answering-based KG.

Lastly, ALFWorld (Shridhar et al., 2020) offers a new dimension by enabling the learning of a general policy in a textual environment and then testing and enhancing it in an embodied environment with a common latent structure. The general tasks also require commonsense reasoning for the agent to make connections between items and attributes, for example, (sink, ‘‘clean’’), (lamp, ‘‘light’’). Likewise, the *attribute* setting of the QAit (Yuan et al., 2019) environment demands agents to understand attribute affordances (cuttable, edible, cookable) to find and alter (cut, eat, cook) objects in the environment.

Safety constraints are often required to ensure that an agent does not make: costly, damaging, and irreparable mistakes when learning (Dulac-Arnold et al., 2019). A common method is to utilize Constrained-MDPs that restrict the agent’s exploration but, defining the restriction requires a specification on which states need to be avoided. Alternatively, Hendrycks et al. (2021) adjust the reward function to avoid bad actions as well as encourage positive actions for more humanistic decision making behaviors. They achieve this by training the agent on a dataset labelled with a numeric scale to denote the morality value of any action, thus providing commonsense knowledge to the agent. The annotations were completed by a group of computer science graduate and undergraduate students over a period of 6 months emphasizing the scale in the challenge of labelling data such as this.

Knowledge Representation Although this is not a challenge of the environment itself, knowledge representation has been a focus of many solutions and therefore we include it here as there are challenges in accurately representing the game’s underlying truth in this way based on the nature of the partially observed state representation.

It can be specified that at any given time step, the game’s state can be represented as a graph

[1] http://www.allthingsjacq.com/interactive_fiction.html#clubfloyd.

that captures observation entities (player, objects, locations, etc) as vertices and the relationship between them as edges. As Text Games are partially observable, an agent can track its belief of the environment into a knowledge graph as it discovers it, eventually converging to an accurate representation of the entire game state (Ammanabrolu and Riedl, 2019).

In contrast to methods that encode their entire KG into a single vector (as shown in Ammanabrolu et al., 2020, and Ammanabrolu et al., 2020a), Xu et al. (2020) suggest an intuitive approach of using multiple sub-graphs with different semantic meanings for multi-step reasoning. Previous approaches have relied on predefined rules and Stanford's Open Information Extraction (Angeli et al., 2015) for deriving information from observations for KG construction. Adhikari et al. (2020) have instead built an agent that is capable of designing and updating it's belief graph without supervision.

Guo et al. (2020) re-frames the Text Games problem by considering observations as passages in a multi-passage RC task. They use an object-centric past-observation retrieval to enhance current state representations with relevant past information and then apply attention to draw focus on correlations for action-value prediction.

2.4 Handicaps

With the challenges introduced, we may now consider the limitations that can be imposed by the generation tools to reduce the complexity of each problem. These handicaps are typically used to limit the scope of the challenges being faced at any one time for more rigorous comparisons and for simplicity.

It has been noted that TextWorld's (Côté et al., 2018) generative functionality explicit advantage is that it can be used to focus on a desired subset of challenges. For example, the **size of the state space** can be controlled and how many commands are required in order to reach the goal. Evaluation of specific generalizability measures can also be improved by controlling the training vs testing variations.

The **partial observability** of the state can also be controlled by augmenting the agent's observations. It is possible for the environment to provide all information about the current game state and therefore reduce the amount an agent must explore to determine the world, relationships, and objects contained within.

Furthermore, the **complexity of the language** itself can be reduced by restricting the agent's vocabulary to in-game words only or the verbs to only those understood by the parser. The grammar can be further simplified by replacing object names with symbolic tokens. It is even possible for the language generation to be avoided completely by converting every generated game into a *choice-based* game where actions at each timestep are defined by a list of pre-defined commands to choose from.

Rewards can simplified with more immediate rewards during training based on the environment state transitions and the known ground truth winning policy rather than simply a sparse reward at the end of a quest as normally provided.

Actions are defined by text commands of at least one word. The interpreter can accept any sequence of characters but will only recognize a tiny subset and moreso only a fraction of these will change the state of the world. The action space is therefore enormous and so two simplifying assumptions are made:

- *Word-level Commands* are sequences of at most L words taken from a fixed vocabulary V.

- *Syntax Commands* have the following structure - verb[noun phrase [adverb phrase]] where [. . .] indicates that the sub-string is optional.

Jericho (Hausknecht et al., 2019a) similarly has a set of possible simplifying steps to reduce the environment's complexity. Most notably, each environment provides agents with the set of valid actions in each game's state. It achieves this by executing a candidate action and looking for the resulting changes to the world-object-tree. To further reduce the difficulty of the games, optional handicaps can be used:

- Fixed random seed to enforce determinism
- Use of load, save functionality
- Use of game-specific templates and vocabulary
- Use of world object tree as an auxiliary state representation or method for detection player location and objects
- Use of world-change-detection to identify valid actions

Name	Task description	Eval	GEN	#Diffic.	max $\#rooms$	max $\#objects$	$\lvert AVS \rvert$	$\overline{len(o_t)}$	$\lvert V \rvert$	max $\lvert quest \rvert$
Zork I (Anderson et al., 1980)	*Collect* the **Twenty Treasures** of Zork	S	N	1	110	251	237	NS	697	396
Treasure Hunter (Côté et al., 2018)	*Collect* **varying items** in a Home	ZS	Y	30	20	2	~ 4	NS	NS	NS
Coin Collector (Yuan et al., 2018)	*Collect* **a coin** in a Home	S,J,ZS	Y	3	90	1	2	64 ± 9	NS	30
FTWP/CookingWorld (Trischler et al., 2019)	*Select & Combine* **varying items** in a Kitchen	J, ZS	N	Various	12	56	18	97 ± 49	20,000	72
Jericho's SoG (Hausknecht et al., 2019a)	A set of classical TGs	S	N	3	NS	221_{avg}	NS	$42, 2_{avg}$	762_{avg}	98_{avg}
QAit (Yuan et al., 2019)	Three QA type settings	ZS	Y	1	12	27	17	93.1	1,647	NS
TW-Home (Ammanabrolu and Riedl, 2019)	*Find* **varying objects** in a home	ZS	Y	2	20	40	NS	94	819	10
TW-Commonsense (Murugesan et al., 2020a)	*Collect* and *Move* **varying items** in a Home	ZS	Y	3	2	7	~ 4	NS	NS	17
TW-Cook (Adhikari et al., 2020)	*Gather* and *process* **cooking ingredients**	J, ZS	Y	5	9	34.1	28.4	NS	NS	3
TextWorld KG (Zelinka et al., 2019)	Constructing KGs from TG observations	–	N	1	N/A	N/A	N/A	29.3	NS	N/A
ClubFloyd (Yao et al., 2020)	Human playthroughs of various TGs	–	N	1	N/A	N/A	NS	NS	39,670	360_{avg}
Jericho-QA (Ammanabrolu et al., 2020a)	QA from context strings	–	N	1	N/A	N/A	N/A	223.2_{avg}	NS	N/A

Table 1: Environments for Textual-Games. Eval. (S)Single, (J) Joint, (ZS) Zero-Shot; GEN: engine support for generation of new games; #Diffic.: number of difficulty settings; $\#rooms$ & $\#objects$: number of rooms and objects per game; AVS: size of Action-Verb Space (NS=Not Specified); $\overline{len(o_t)}$: mean number of tokens in the observation o_t; $|V|$: Vocabulary size; and $|quest|$: length of optimal trajectory.

Jericho also introduces a set of possible restrictions to the action-space.

- *Template-based* action spaces separate the problem of picking actions into two (i) picking a template (e.g., ''take _ from _''); (ii) filling the template with objects (e.g., ''apple'', ''fridge''). Essentially this reduces the issue to verb selection and contextualised entity extraction.
- *Parser-Based* action spaces require the agent to generate a command word-per-word, sometimes following a pre-specified structures similar to (verb, object_1, modifier, object_2).
- *Choice-based* requires agents to rank predefined set of actions without any option for ''creativity'' from the model itself.

Lastly, the observation space may be enchanced with the outputs of bonus commands such as ''look'' and ''inventory''. These are commands that the agent can issue on its own but are not considered an actual step in the exploration process that could be costly and produce risks in the real-world when asked for.

3 Benchmark Environments and Agents

Thus far, the majority of researchers have independently generated their own environments with TextWorld (Côté et al., 2018). As the field moves towards more uniformity in evaluation, a clear overview of which environments are already generated, their design goals and the benchmark approach is needed.

Table 1 shows the publicly available environments and datasets. Much of the recent research has been published within the past few years (2018–2020). Jericho's Suite of Games (SoG) (Hausknecht et al., 2019a) is a collection of 52 games and therefore has its results averaged across all games included in evaluation.

We find that many of the environments focus on exploring increasingly complex environments to *'collect'* an item of some kind ranging from a simple coin to household objects. This is due to the TextWorld generator's well defined logic rules for such objects but can also be a limitation on the possible scope of the environment.

When evaluating an agent's performance, three types of evaluation settings are typically considered:

- **Single Game** evaluate agents in the same game under the same conditions,
- **Joint** settings evaluate agents trained on the same *set* of games that typically share some similarities in the states seen,
- **Zero-Shot** settings evaluate agents on games completely unseen in training.

The difficulty settings of the environments are defined by the complexity of the challenges that the agents are required to overcome. In most cases, these have been limited to just a few levels. However, CookingWorld defines the challenge by a set of key variables that can be changed separately or in combination and therefore does not offer clear discrete difficulty settings.

The max number of rooms and objects depends heavily on the type of task. Coin Collecter, for example, has only 1 object to find as the complexity comes from traversing a number of rooms. Alternatively, CookingWorld has a limited number

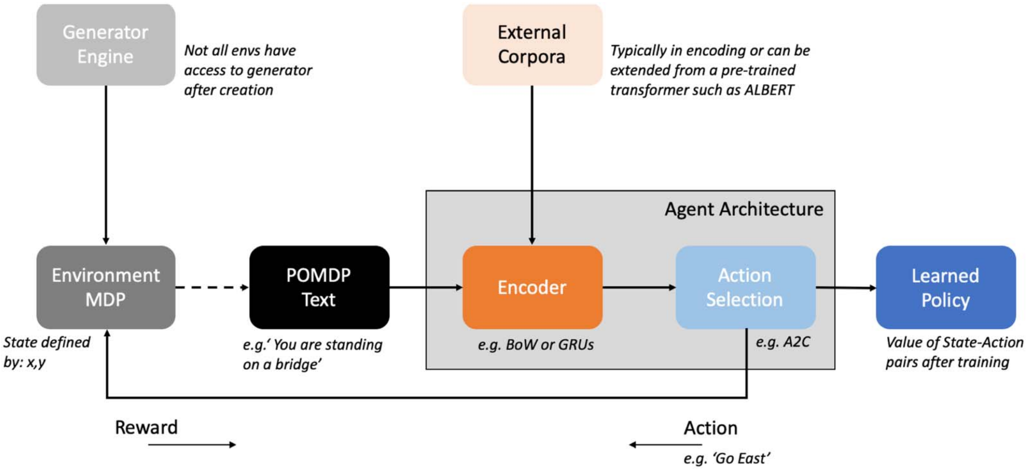

Generator Engine
Not all envs have access to generator after creation
External Corpora
Typically in encoding or can be extended from a pre-trained transformer such as ALBERT
Agent Architecture
Environment MDP
POMDP Text
Encoder
Action Selection
Learned Policy
State defined by: x,y
e.g.' You are standing on a bridge'
e.g. BoW or GRUs
e.g. A2C
Value of State-Action pairs after training
Reward
Action
e.g. 'Go East'

Figure 2: Overview of the Architecture Structure of Agents Applied to a Simple Text Game Example.

Name	Encoder	Action Selector	KG	PTF	Pre-Training	AS	Tasks
Ammanabrolu et al. (2020a)	GRU	A2C	DL	ALBERT	J-QA	TB	Zork1
Xu et al. (2020)	GRU	A2C	DL	none	none	TB	JSoG
Ammanabrolu and Hausknecht (2020)	GRU	A2C	DL	none	ClubFloyd	TB	JSoG
Murugesan et al. (2020b)	GRU	A2C	DL+CS	none	GloVe	CB	TW-Commonsense
Yao et al. (2020)	GRU	DRRN	none	GPT-2	ClubFloyd	CB	JSoG
Adolphs and Hofmann (2020)	Bi-GRU	A2C	none	none	TS, GloVe$_{100}$	TB	CW
Guo et al. (2020)	Bi-GRU	DQN	none	none	GloVe$_{100}$	TB	JSoG
Xu et al. (2020)	TF	DRRN	none	none	none	TB	JSoG
Yin and May (2020)	TF,LSTM	DSQN	none	none	none	NS	CW, TH
Adhikari et al. (2020)	R-GCN, TF	DDQN	DL	none	TS	CB	TW-Cook
Zahavy et al. (2018)	CNN	DQN	none	none	word2vec$_{300}$	CB	Zork1
Ammanabrolu and Riedl (2019)	LSTM	DQN	DL	none	TS, GloVe$_{100}$	CB	TW-Home
He et al. (2016)	BoW	DRRN	none	none	none	CB	other
Narasimhan et al. (2015)	LSTM	DQN	none	none	none	PB	other
Yin et al. (2020)	BERT	DQN	DL	BERT	none	CB	FTWP, TH
Madotto et al. (2020)	LSTM	Seq2Seq	none	none	GloVe$_{100,300}$	PB	CC, CW

Table 2: Overview of recent architectural trends. ENC, state/action encoder; KG, knowledge graph (DL: dynamically learned; CS: commonsense); PTF, pretrained Transformer; PreTr, pretraining (TS: task specific); AS, Action space (TB: template-based; PB: parser based; CB: choice-based).

of rooms and instead focuses on a large number of objects to consider. Zork is naturally the most complex in this regards as an adaptation of a game designed for human players. Likewise, the complexity of the vocabulary depends heavily on the task but it is clear to see that the environments limit the number of Action-Verbs for simplicity.

3.1 Agent Architectures

A Deep-RL agent's architecture (see Figure 2) consists of two core components: (i) state encoder and (ii) and action scorer (Mnih et al., 2013). The first is used to encode game information such as observations, game feedback, and KG representations into state approximations. The encoded information is then used by an action selection agent to estimate the value of actions in each state.

Table 2 provides an overview of recent architectural trends for comparison. We find that the initial papers in 2015 used the standard approaches of LSTM or Bag of Words for the encoder and a Deep Q-Network (DQN) for the action selector. More recent developments have been experimenting with both parts towards improved results. Notably, a range of approaches for the encoding have been introduced, with Gated Recurrent Units (GRUs) having become the most common in 2020. Note that there have been fewer variations in the choice of action selector where either Actor-Critic (A2C) or a DQN is typically used. Furthermore, the use of KGs and Pre-trained Transformers is limited and many of the works that use these were published in 2020.

Most of the agents were applied to either TextWorld/CookingWorld or Jericho. We typically find that alternative environments align to consistent setups; either the authors create a new game that mimics simple real-world rules (similar to TextWorld) or they apply their methods to well know pre-existing games (similar to Jericho). Specifically, Narasimhan et al. (2015) used two self generated games themselves: 'Home World' to mimic the environment of a typical house and 'Fantasy World' that is more challenging and akin to a role-playing game. Alternatively, He et al. (2016) used a deterministic Text Game 'Saving John' and and a larger-scale stochastic Text Game 'Machine of Death', both pre-existing from a public library.

Encoders used include both simplistic state encodings in the form of Bag of Words (BoW), but also recurrent modules like: GRU (Cho et al., 2014), Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), Transformer (TF) (Vaswani et al., 2017), and Relational Graph Convolutional Network (R-GCN) (Schlichtkrull et al., 2018). Recently, Advantage Actor Critic (A2C) (Mnih et al., 2016) has gained popularity for the action selection method, with variants of the Deep Q-Network (Mnih et al., 2013), such as the Deep Reinforcement Relevance Network (DRRN) (He et al., 2016), Double DQN (DDQN) (Hasselt et al., 2016), and Deep Siamese Q-Network (DSQN) (Yin and May, 2020).

Task Specific Pre-Training entails heuristically establishing a setting in which a submodule learns priors before interacting with training data. For example, Adolphs and Hofmann (2020) pretrain on a collection of food items to improve generalizability to unseen objects and Adhikari et al. (2020) pretrain a KG constructor on trajectories of similar games the agent is trained on. Chaudhury et al. (2020) showed that training an agent on pruned observation space, where the semantically least relevant tokens are removed in each episode to improve generalizability to unseen domains whilst also improving the sample efficiency due to requiring less training games. Furthermore, Jain et al. (2020) propose learning different action-value functions for all possible scores in a game, thus effectively learning separate value functions for each subtask of a whole.

Lastly, the **action-space** typically varies between template- or choice-based depending on the type of task. Only two papers have considered a parser based approach: Narasimhan et al. (2015) and Madotto et al. (2020).

3.2 Benchmark Results

The following section summarizes some of the results published thus far as a means to review the performance of the baselines as they are presented by the original papers to be used for future comparisons.

Treasure Hunter was introduced by Côté et al. (2018) as part of the TextWorld formalization and inspired by a classic problem to navigate a maze to find a specific object. The agent and objects are placed in a randomly generated map. A coloured object near the agent's start location provides an indicator of which object to obtain provided in the welcome message. A straightforward reward signal is defined as positive for obtaining the correct object and negative for an incorrect object with a limited number of turns available.

Increasing difficulties are defined by the number of rooms, quest length, and number of locked doors and containers. Levels 1 to 10 have only 5 rooms, no doors, and an increasing quest length from 1 to 5. Levels 11 to 20 have 10 rooms, include doors and containers that may need to be opened, and quest length increasing from 2 to 10. Lastly, levels 21 to 30 have 20 rooms, locked doors and containers that may need to be unlocked and opened, and quest length increasing from 3 to 20.

For the evaluation, two state-of-the-art agents (BYU [Fulda et al., 2017] and Golovin [Kostka et al., 2017]) were compared to a choice-based random agent in a *zero-shot* evaluation setting. For completeness, each was applied to 100 generated games at varying difficulty levels up to a maximum of 1,000 steps. The results of this are shown in Table 3, where we note that the random agent performs best but this is due to the choice-based method removing the complexity of the compositional properties of language. The authors noted that may not be directly comparable to the other agents but provides an indication of the difficulty of the tasks.

CookingWorld, the second well-established environment developed using TextWorld by Trischler et al. (2019), is used in *joint* and *zero-shot* settings, which both enable testing for generalization. The former entails training and evaluating the agent on the same set of games, while the latter uses an unseen test set upon evaluation. The fixed dataset provides 4,400 training, 222 validation, and 514 test games with 222 different types of games in various game difficulty

	Random		BYU		Golovin	
Difficulty	Avg. Score	Avg. Steps	Avg. Score	Avg. Steps	Avg. Score	Avg. Steps
level 1	0.35	9.85	0.75	85.18	0.78	18.16
level 5	−0.16	19.43	−0.33	988.72	−0.35	135.67
level 10	−0.14	20.74	−0.04	1000	−0.05	609.16
level 11	0.30	43.75	0.02	992.10	0.04	830.45
level 15	0.27	63.78	0.01	998	0.03	874.32
level 20	0.21	74.80	0.02	962.27	0.04	907.67
level 21	0.39	91.15	0.04	952.78	0.09	928.83
level 25	0.26	101.67	0.00	974.14	0.04	931.57
level 30	0.26	108.38	0.04	927.37	0.74	918.88

Table 3: Results of agents applied to Treasure Hunter's one-life tasks (Côté et al., 2018).

	DSQN	LeDeepChef	Go-Explore Seq2Seq	BERT-NLU-SE	LSTM-DQN	DRRN
Single	–	–	88.1%	–	52.1%	80.9%
Joint	–	–	56.2%	–	3.1%	22.9%
Zero-Shot	58%	69.3%	51%	77%	2%	22.2%
Treasure Hunter (OoD)	42%	–	–	57%	–	–
% won games (Zero-Shot)	–	–	46.6%	71%	3.2%	8.3%
avg #steps (Zero-Shot)	–/ 100	43.9 / 100	24.3 / 50	–/ 100	48.5 / 50	38.8 / 50
#training steps	10^7	13,200	NS	$10^7_{Tchr}, 5 * 10^5_{stdn}$	NS	NS
Admissable Actions	NS	N	N	Y	Y	Y
Curriculum Learning	N	N	Y	Y	N	N
Imitation Learning	N	N	Y	Y	N	N
Training time	NS	NS	NS	Tchr: 35 days, Stdn: 5 days	NS	NS

Table 4: Results on the public CookingWorld dataset. OoD, out of domain.

settings; the current best results are shown in Table 4. However, there are some variances in the split of training and test games, making it challenging to compare the results. Most models were trained using the games split of 4,400/514 specified before, but some split it as 3,960/440 and some do not state the split at all. Table 4 shows the results of this comparison with a percentage score relative to the maximum reward.

Jericho's Suite of Games was defined by Hausknecht et al. (2019a) and has been used for testing agents in a *single game* setting. The suite provides 57 games, of which a variation of 32 are used due to excessive complexity, therefore the agents are initialized, trained, and evaluated on each of them separately. Agents are assessed over the *final score*, which is, by convention, averaged over the last 100 episodes (which in turn is usually averaged over 5 agents with different initialization seeds). The current state of the art results on the Jericho game-set can be seen in Table 5 where numeric results are shown relative to the Maximum Reward with the aggregated average percentage shown in the final row.[2]

The difficulty of each game has been summarized in full by Hausknecht et al. (2019a) but not included within this paper due to its size. The authors categorize difficulty with the features of template action space size, solution length, average steps per reward, stochastic, dialog, darkness, nonstandard actions, and inventory limit. However, it has been noted that some games have specific challenges not captured by these alone that make them more difficult. For example "*9:05 poses a difficult exploration problem as this game features only a single terminal reward indicating success or failure at the end of the episode*".

4 Opportunities for Future Work

The previous sections summarized the generation tools for Text Games, the subsequent environments

[2] NAIL is a rule-based agent and is noted to emphasize the capabilities of learning based models.

	MaxR	CALM-DRRN	SHA-KG	Trans-v-DRRN	MPRC-DQN	KG-A2C	TDQN	DRRN	NAIL	$\|T\|$	$\|V\|$
905	1	**0**	–	**0**	**0**	**0**	**0**	**0**	**0**	82	296
acorncourt	30	0	1.6	**10**	**10**	0.3	1.6	**10**	0	151	343
advent	350	36	–	–	**63.9**	36	36	20.6	36	189	786
advland	100	0	–	25.6	**42.2**	0	0	20.6	0	156	398
affilicted	75	–	–	2.0	**8.0**	–	1.4	2.6	0	146	762
anchor	100	**0**	–	–	**0**	**0**	**0**	**0**	**0**	260	2257
awaken	50	**0**	–	–	**0**	**0**	**0**	**0**	**0**	159	505
balances	51	9.1	**10.0**	–	**10**	**10**	4.8	**10**	10	156	452
deephome	300	1	–	–	1	1	1	1	**13.3**	173	760
detective	360	289.7	208.0	288.8	**317.7**	207.9	169	197.8	136.9	197	344
dragon	25	0.1	0.2	–	0.04	0	−5.3	−3.5	**0.6**	177	1049
enchanter	400	19.1	**20**	**20**	**20**	1.1	8.6	**20**	0	290	722
gold	100	–	–	–	0	–	**4.1**	0	3	200	728
inhumane	90	**25.7**	5.4	–	0	3	0.7	0	0.6	141	409
jewel	90	0.3	1.8	–	**4.46**	1.8	0	1.6	1.6	161	657
karn	170	2.3	–	–	**10**	0	0.7	2.1	1.2	178	615
library	30	9.0	15.8	17	**17.7**	14.3	6.3	17	0.9	173	510
ludicorp	150	10.1	17.8	16	**19.7**	17.8	6	13.8	8.4	187	503
moonlit	1	**0**	–	–	**0**	**0**	**0**	**0**	**0**	166	669
omniquest	50	6.9	–	–	10	3	**16.8**	5	5.6	207	460
pentari	70	0	**51.3**	34.5	44.4	50.7	17.4	27.2	0	155	472
reverb	50	–	10.6	**10.7**	2.0	–	0.3	8.2	0	183	526
snacktime	50	**19.4**	–	–	0	0	9.7	0	0	201	468
sorcerer	400	6.2	29.4	–	**38.6**	5.8	5	20.8	5	288	1013
spellbrkr	600	**40**	**40**	**40**	25	21.3	18.7	37.8	**40**	333	844
spirit	250	1.4	**3.8**	–	**3.8**	1.3	0.6	0.8	1	169	1112
temple	35	0	7.9	7.9	**8.0**	7.6	7.9	7.4	7.3	175	622
tryst205	350	–	6.9	9.6	**10**	–	0	9.6	2	197	871
yomomma	35 –	–	–	–	**1**	–	0	0.4	0	141	619
zenon	20	0	**3.9**	–	0	**3.9**	0	0	0	149	401
zork1	350	30.4	34.5	36.4	**38.3**	34	9.9	32.6	10.3	237	697
zork3	7	0.5	0.7	0.19	**3.63**	0.1	0	0.5	1.8	214	564
ztuu	100	3.7	25.2	4.8	**85.4**	9.2	4.9	21.6	0	186	607
Best Agent % / count		21.2% / 7	18.2% / 6	15.2% / 5	69.7% / 23	18.2% / 6	18.2% / 6	21.2% / 7	21.2% / 7		
Avg. norm / # games		9.4% / 28	19.6% / 20	22.3% / 15	17% / 33	10.8% / 28	6% / 33	10.7% / 32	4.8% / 33		
Handicaps		$\{1\}$	$\{1, 2, 4\}$	NS	$\{1, 2, 4\}$	$\{1, 2, 4\}$	$\{1, 2, 4\}$	$\{1, 4\}$	N/A		
Train steps		10^6	10^6	10^5	10^5	1.6×10^6	10^6	1.6×10^6	N/A		
Train time		NS	NS	NS	8h-30h per game	NS	NS	NS	N/A		

Table 5: The current state of the art results on the Jericho game-set. Blue: likely to be solved in near future. Orange: progress is likely, significant scientific progress necessary to solve. Red: very difficult even for humans, unthinkable for current RL. MaxR, Maximum possible reward; $|T|$, number of templates (e.g., put the __ in __); $|V|$, size of vocabulary set.

created thus far, and the architectures posed as solutions to these problems. The most clear improvement opportunities come from an approach that directly addresses the challenges that are yet to be well solved and are given in detail alongside the current benchmark environments and solutions in Section 2.3.

The primary challenge of partial observability exists in all Text Games, with some solutions using handcrafted reward functions (Yuan et al., 2018) or knowledge graphs to leverage past information (Ammanabrolu and Hausknecht, 2020). Additionally, many recent works address this challenge in their agent architecture by utilizing transformer models pre-trained on natural language corpora to introduce background knowledge, thus reducing the amount that needs to be observed directly. This has been supported by work published in the NLP community with methods such as BERT (Yin et al., 2020), GPT-2 (Yao et al., 2020), and ALBERT (Ammanabrolu et al., 2020a) and continued developments from this community will support the advancements of future agent's architectures.

One of the most interesting challenges is how agents learn to use commonsense reasoning about the problems and whether this can be used to generalize across tasks, other games, and even into the real world. For example, an agent that can learn how a key is used in a Text Game setting would have reasoning beyond any agent simply trained to color match keys based on reward signals alone (e.g., a robot using visual inputs). This has been noted as an important part of the planning stage and could further take advantage of work on dynamically learned knowledge graphs that have become common in recent works (such as Das et al., 2019). Specifically, this enables the use of pre-trained knowledge graphs on readily available text corpora alongside the commonsense understanding of previously seen tasks before training on a new environment on which it may be expensive to collect data (e.g., implementing a robot into a new user's home). However, language acquisition in these environments is still an open problem but with developments on Urbanek et al.'s (2019) environment specifically designed for grounding language this will become closer to being solved.

Another benefit of language understanding and knowledge graphs (particularly in the planning stage) is that they can used for interpretability. This is not a challenge unique to Text Games or Reinforcement Learning, as calls for research have been made in similar machine learning do mains (Chu et al., 2020). A notable methodology that is worth considering is Local Interpretable Model-Agnostic Explanations (LIME), introduced by Ribeiro et al. (2016). Recent work has been published that specifically considers interpretability for RL (Peng et al., 2021) whereby the authors designed their agent to 'think out loud' as it was making decisions on Text Games and evaluated with human participants.

Lastly, from the contributions analyzed in this survey, only 5 papers report the amount of time and resources their methods needed for training. Continuing the trend of publishing these specifications is essential in making results more reproducible and applicable in applied settings. Reducing the amount of resources and training time required as a primary motive allows for the domain to be practical as a solution and also accessible given that not all problems allow for unlimited training samples.

5 Conclusion

Many of the challenges faced within Text Games also exist in other language-based Reinforcement Learning problems, highlighted by Luketina et al. (2019). Likewise, these challenges also exist in 'real-world' settings noted by Dulac-Arnold et al. (2019) including: limited data, safety constraints, and/or a need for interpretability.

We find that most Text Games methodologies focus on either an agent's ability to learn efficiently or generalize knowledge. A continued development in this direction is the primary motive for much of the recent research and could lead to solutions that work sufficiently well on these real-world problem settings for the primary challenge of limited and/or expensive data. Likewise, specific work on interpretability (such as that of Peng et al., 2021) and language acquisition may transfer to problems that interact in the real-world directly with humans.

To this end, this study has summarized recent research developments for interactive fiction games that have generated a set of environments and tested their architectures for generalization and overfitting capabilities. With the comparisons made in this work and a uniform set of environments and baselines, new architectures can be developed and then systematically evaluated for improving results within Text Games and subsequently other Reinforcement Learning problems that include language.

References

Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and Will Hamilton. 2020. Learning dynamic belief graphs to generalize on text-based games. *Advances in Neural Information Processing Systems*, 33.

Leonard Adolphs and T. Hofmann. 2020. Ledeepchef: Deep reinforcement learning agent for families of text-based games. In *AAAI*. https://doi.org/10.1609/aaai.v34i05.6228

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

Prithviraj Ammanabrolu and Mark O. Riedl. 2019. Playing text-adventure games with graph-based deep reinforcement learning. *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. https://doi.org/10.18653/v1/N19-1358

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

Prithviraj Ammanabrolu, Jack Urbanek, Margaret Li, Arthur Szlam, Tim Rocktaschel, and J. Weston. 2020b. How to motivate your dragon: Teaching goal-driven agents to speak and act in fantasy worlds. *ArXiv*, abs/2010.00685. https://doi.org/10.18653/v1/2021.naacl-main.64

Tim Anderson, Marc Blank, Bruce Daniels, and Dave Lebling. 1980. Zork: The great underground empire – part i.

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, Beijing, China. Association for Computational Linguistics. `https://doi.org/10.3115/v1/P15-1034`

Subhajit Chaudhury, Daiki Kimura, Kartik Talamadupula, Michiaki Tatsubori, Asim Munawar, and Ryuki Tachibana. 2020. Bootstrapped q-learning with context relevant observation pruning to generalize in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, Online, pages 3002–3008. `https://doi.org/10.18653/v1/2020.emnlp-main.241`

Kyunghyun Cho, B. V. Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Empirical Methods in Natural Language Processing*, abs/1406.1078.

Eric Chu, Nabeel Gillani, and Sneha Priscilla Makini. 2020. Games for fairness and interpretability. In *Companion Proceedings of the Web Conference 2020*, WWW '20, pages 520–524, New York, NY, USA. Association for Computing Machinery. `https://doi.org/10.1145/3366424.3384374`

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, J. Moore, Matthew J. Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. *Workshop on Computer Games*, pages 41–75. `https://doi.org/10.1007/978-3-030-24337-1_3`

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2019. Building dynamic knowledge graphs from text using machine reading comprehension. *ICLR*.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. 2019. Challenges of real-world reinforcement learning. *Proceedings of the 36th International Conference on Machine Learning*.

Nancy Fulda, Daniel Ricks, Ben Murdoch, and David Wingate. 2017. What can you do with a rock? Affordance extraction via word embeddings. *arXiv preprint arXiv: 1703.03429*. `https://doi.org/10.24963/ijcai.2017/144`

Xiaoxiao Guo, M. Yu, Yupeng Gao, Chuang Gan, Murray Campbell, and S. Chang. 2020. Interactive fiction game playing as multi-paragraph reading comprehension with reinforcement learning. *EMNLP*.

H. V. Hasselt, A. Guez, and D. Silver. 2016. Deep reinforcement learning with double q-learning. *AAAI*. `https://doi.org/10.1609/aaai.v30i1.10295`

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

Matthew Hausknecht, Ricky Loynd, Greg Yang, Adith Swaminathan, and Jason D. Williams. 2019b. Nail: A general interactive fiction agent. *arXiv preprint arXiv:1902.04259*.

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. *Association for Computational Linguistics (ACL)*. `https://doi.org/10.18653/v1/P16-1153`

Dan Hendrycks, Mantas Mazeika, Andy Zou, Sahil Patel, Christine Zhu, Jesus Navarro, Dawn Song, Bo Li, and Jacob Steinhardt. 2021. What would jiminy cricket do? Towards agents that behave morally. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780. `https://doi.org/10.1162/neco.1997.9.8.1735`

Vishal Jain, William Fedus, Hugo Larochelle, Doina Precup, and Marc G. Bellemare. 2020.

Algorithmic improvements for deep reinforcement learning applied to interactive fiction. In *AAAI*, pages 4328–4336. https://doi.org/10.1609/aaai.v34i04.5857

Bartosz Kostka, Jaroslaw Kwiecieli, Jakub Kowalski, and Pawel Rychlikowski. 2017. Text-based adventures of the Golovin AI agent. *Computational Intelligence and Games (CIG)*, pages 181–188. https://doi.org/10.1109/CIG.2017.8080433

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. 2019. A survey of reinforcement learning informed by natural language. *arXiv e-prints*. https://doi.org/10.24963/ijcai.2019/880

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. *IJCAI*. https://doi.org/10.24963/ijcai.2020/207

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning, PMLR 2016*, pages 1928–1937.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with deep reinforcement learning. *CoRR*, abs/1312.5602.

Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Pushkar Shukla, Sadhana Kumaravel, Gerald Tesauro, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. 2020a. Text-based RL agents with commonsense knowledge: New challenges, environments and baselines. *ArXiv*, abs/2010.03790.

Keerthiram Murugesan, Mattia Atzeni, Pushkar Shukla, Mrinmaya Sachan, Pavan Kapanipathi, and Kartik Talamadupula. 2020b. Enhancing text-based reinforcement learning agents with commonsense knowledge. *arXiv preprint arXiv:2005.00811*.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. *EMNLP*. https://doi.org/10.18653/v1/D15-1001

Xiangyu Peng, Mark O. Riedl, and Prithviraj Ammanabrolu. 2021. Inherently explainable reinforcement learning in natural language. *ArXiv*, abs/2112.08907.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. ''why should I trust you?'': Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144. https://doi.org/10.1145/2939672.2939778

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling, Springer. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. https://doi.org/10.1007/978-3-319-93417-4_38

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *ArXiv*, abs/2010.03768.

D. Silver, Aja Huang, Chris J. Maddison, A. Guez, L. Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, S. Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and Demis Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489. https://doi.org/10.1038/nature16961

Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning*, 1st edition. MIT Press, Cambridge, MA, USA.

Ruo Yu Tao, Marc-Alexandre Côté, Xingdi Yuan, and Layla El Asri. 2018. Towards solving text-based games by producing adaptive action spaces. *ArXiv*, abs/1812.00855.

Adam Trischler, Marc-Alexandre Côté, and Pedro Lima. 2019. First textworld problems, the competition: Using text-based games to advance capabilities of ai agents.

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and J. Weston. 2019. Learning to speak and act in a fantasy text adventure game. *ArXiv*, abs/1903.03094. https://doi.org/10.18653.18653/v1/D19-1062

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008.

Y. Xu, L. Chen, M. Fang, Y. Wang, and C. Zhang. 2020. Deep reinforcement learning with transformers for text adventure games. In *2020 IEEE Conference on Games (CoG)*, pages 65–72. https://doi.org/10.1109/CoG47356.2020.9231622

Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, Joey Tianyi Zhou, and Chengqi Zhang. 2020. Deep reinforcement learning with stacked hierarchical attention for text-based games. *Advances in Neural Information Processing Systems*, 33.

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)*.

Xusen Yin and Jonathan May. 2020. Zero-shot learning of text adventure games with sentence-level semantics. *arXiv preprint arXiv:2004.02986*.

Xusen Yin, R. Weischedel, and Jonathan May. 2020. Learning to generalize for sequential decision making. *EMNLP*. https://doi.org/10.18653/v1/2020.findings-emnlp.273

Xingdi Yuan, Marc-Alexandre Côté, Jie Fu, Zhouhan Lin, Christopher Pal, Yoshua Bengio, and Adam Trischler. 2019. Interactive language learning by question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Join Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2796–2813. https://doi.org/10.18653/v1/D19-1280

Xingdi Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Remi Tachet des Combes, Matthew Hausknecht, and Adam Trischler. 2018. Counting to explore and generalize in text-based games. *35th International Conference on Machine Learning, Exploration in Reinforcement Learning Workshop*.

Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J. Mankowitz, and Shie Mannor. 2018. Learn what not to learn: Action elimination with deep reinforcement learning. In *Advances in Neural Information Processing Systems 2018*, pages 3562–3573.

Mikulá Zelinka, Xingdi Yuan, Marc-Alexandre Côté, R. Laroche, and Adam Trischler. 2019. Building dynamic knowledge graphs from text-based games. *ArXiv*, abs/1910.09532.