Interactive Fiction Games: A Colossal Adventure

Matthew Hausknecht
Microsoft Research AI

Prithviraj Ammanabrolu
Georgia Institute of Technology

Marc-Alexandre Côté
Microsoft Research Montréal

Xingdi Yuan
Microsoft Research Montréal

Abstract

A hallmark of human intelligence is the ability to understand and communicate with language. Interactive Fiction games are fully text-based simulation environments where a player issues text commands to effect change in the environment and progress through the story. We argue that IF games are an excellent testbed for studying language-based autonomous agents. In particular, IF games combine challenges of combinatorial action spaces, language understanding, and commonsense reasoning. To facilitate rapid development of language-based agents, we introduce Jericho, a learning environment for man-made IF games and conduct a comprehensive study of text-agents across a rich set of games, highlighting directions in which agents can improve.

1 Introduction

Interactive fiction (IF) games are software environments in which players observe textual descriptions of the simulated world, issue text actions, and receive score progress through the story. As illustrated by Figure 1, humans bring competencies in natural language understanding, commonsense reasoning, and deduction to bear in order to infer the context and objectives of a game. We believe IF environments will provide a good testbed for studying artificial development of these capabilities. Beyond games, real-world applications such as voice-activated personal assistants also exist at the intersection of natural language understanding, natural language generation, and sequential decision making.

From a machine learning perspective, Interactive Fiction games exist at the intersection of natural language processing and sequential decision making. Like many NLP tasks, they require natural language understanding, but unlike most NLP tasks, IF games are sequential decision making problems in which actions change the subsequent world states of the game and choices made early in a game may have long term effects on the eventual endings. Reinforcement Learning (Sutton and Barto 1998) studies sequential decision making problems and has shown promise in vision-based (Jaderberg et al. 2016) and control-based (OpenAI et al. 2018) environments, but has less commonly been applied in the context of language-based tasks.

Observation: This bedroom is extremely spare, with dirty laundry scattered haphazardly all over the floor. Cleaner clothing can be found in the dresser. A bathroom lies to the south, while a door to the east leads to the living room. On the end table are a telephone, a wallet and some keys.
The phone rings.

Action: **Answer phone**

Observation: You pick up the phone. "Hadley!" a shrill voice cries. "Hadley, haven't you even left yet?? You knew that our presentation was at nine o'clock sharp! First the thing with the printers, now this - there won't even be enough left of you for Bowman to fire once he's done with you. Now get the hell down here!!"

Action: **Examine me**

Observation: You're covered with mud and dried sweat. It was an exhausting night - no wonder you overslept! Even making it to the bed before conking out was a heroic accomplishment.

Figure 1: Excerpt from Adam Cadre's 9:05.

Why study environments that feature ungrounded natural language instead of multimodal environments that provide visual grounding for language? We believe that ungrounded natural language is inescapable in certain real world tasks such as voice-activated personal assistants.

The contributions of this paper are as follows: First, we introduce Jericho, a learning environment for human-made IF games. Second, we introduce a template-based action space and that we argue is appropriate for language generation. Third, we conduct an empirical evaluation of learning agents across a set of human-made games.

2 Research Challenges

From the perspective of reinforcement learning, IF games can be modeled as partially observable Markov decision processes (POMDPs) defined by (S, T, A, O, R). Observations $o \in O$ correspond to the game's text responses, while latent states $s \in S$ correspond to player and item locations, inventory contents, monsters, etc. Text-based actions $a \in A$ change the game state according to an latent transition function $T(s'|s, a)$, and the agent receives rewards r from an unknown reward function $R(s, a)$. To succeed in these environments, agents must generate natural language actions, reason about entities and affordances, and represent their knowledge about the game world. We present these challenges in greater detail:

Combinatorial Action Space Reinforcement learning has studied agents that operate in discrete or continuous action space environments. However, IF games require the agent to operate in the combinatorial action space of natural language. Combinatorial spaces pose extremely difficult exploration problems for existing agents. For example, an agent generating a four-word sentence from a modest vocabulary of size 700, is effectively exploring a space of $|700^4| = 240$ billion possible actions. Further complicating, this challenge is the fact that natural language commands are interpreted by the game's parser - which recognizes a game-specific subset of possible commands. For example, out of the 240 billion possible actions there may be only ten *valid actions* at each step - actions that are both recognized by the game's parser and generate a change in world state.

As discussed in Section 4.1, we propose the use of a template-based action space in which the agent first chooses from a template of the form *put OBJ in OBJ* and then fills in the blanks using the vocabulary. A typical game may have around 200 templates each with up to two blanks - yielding an action space $|200 * 700^2| = 98$ million, three orders of magnitude smaller than the naive space but six orders of magnitude greater than most discrete RL environments.

Commonsense Reasoning Due to the lack of graphics, IF games rely on the player's commonsense knowledge as a prior for how to interact with the game world. For example, a human player encountering a locked chest intuitively understands that the chest needs to be unlocked with some type of key, and once unlocked, the chest can be opened and will probably contain useful items. They may make a mental note to return to the chest if a key is found in the future. They may even mark the location of the chest on a map to easily find their way back.

These inferences are possible for humans who have years of embodied experience interacting with chests, cabinets, safes, and all variety of objects. Artificial agents lack the commonsense knowledge gained through years of grounded language acquisition and have no reason to prefer opening a chest to eating it. Also known as *affordance extraction* (Gibson 1977; Fulda et al. 2017), the problem of choosing which actions or verbs to pair with a particular noun is central to IF game playing. However, the problem of commonsense reasoning extends much further than affordance extraction: Games require planning which items to carry in a limited inventory space, strategically deciding whether to fight or flee from a monster, and spatial reasoning such as stacking garbage cans against the wall of a building to access a second-floor window.

Knowledge Representation IF games span many distinct locations, each with unique descriptions, objects, and characters. Players move between locations by issuing navigational commands like *go west*. Due to the large number of locations in many games, humans often create maps to navigate efficiently and avoid getting lost. This gives rise to the **Textual-SLAM** problem, a textual variant of Simultaneous localization and mapping (SLAM) (Thrun, Burgard, and Fox 2005) problem of constructing a map while navigating a new environment. In particular, because connectivity between locations is not necessarily Euclidean, agents need to be able to detect when a navigational action has succeeded or failed and whether the location reached was previously seen or new. Beyond location connectivity, it's also helpful to keep track of the objects present at each location, with the understanding that objects can be nested inside of other objects, such as food in a refrigerator or a sword in a chest.

3 Related Work

In contrast to the *parser-based games* studied in this paper, *choice-based games* provide a list of possible actions at each step, so learning agents must only choose between the candidates. The DRRN algorithm for choice-based games (He et al. 2016; Zelinka 2018) estimates Q-Values for a particular action from a particular state. This network is evaluated once for each possible action, and the action with the maximum Q-Value is selected. While this approach is effective for choice-based games which have only a handful of candidate actions at each step, it is difficult to scale to parser-based games where the action space is vastly larger.

In terms of *parser-based games*, such as the ones examined in this paper, several approaches have been investigated: LSTM-DQN (Narasimhan, Kulkarni, and Barzilay 2015), considers *verb-noun* actions up to two-words in length. Separate Q-Value estimates are produced for each possible verb and object, and the action consists of pairing the maximally valued verb combined with the maximally valued object. LSTM-DQN was demonstrated to work on two small-scale domains, but human-made games, such as those studied in this paper, represent a significant increase in both complexity and vocabulary.

Another approach to affordance extraction (Fulda et al. 2017) identified a vector in word2vec (Mikolov et al. 2013) space that encodes affordant behavior. When applied to the noun *sword*, this vector produces affordant verbs such as *vanquish*, *impale*, *duel*, and *battle*. The authors use this method to prioritize verbs for a Q-Learning agent to pair with in-game objects.

An alternative strategy has been to reduce the combinatorial action space of parser-based games into a discrete space containing the minimum set of actions required to finish the game. This approach requires a walkthrough or expert demonstration in order to define the space of minimal actions, which limits its applicability to new and unseen games. Following this approach, (Zahavy et al. 2018) employ this strategy with their action-elimination network, a

classifier that predicts which predefined actions will not effect any world change or be recognized by the parser. Masking these invalid actions, the learning agent subsequently evaluates the set of remaining valid actions and picks the one with the highest predicted Q-Value.

The TextWorld framework (Côté et al. 2018) supports procedural generation of parser-based IF games, allowing complexity and content of the generated games to be scaled to the difficulty needed for research. TextWorld domains have already proven suitable for reinforcement learning agents (Yuan et al. 2018) which were shown to be capable of learning on a set of environments and then generalizing to unseen ones at test time. Recently, Yuan et al. (2019) propose QAit, a set of question answering tasks based on games generated using TextWorld. QAit focuses on helping agents to learn procedural knowledge in an information-seeking fashion, it also introduces the practice of generating unlimited training games on the fly. With the ability to scale the difficulty of domains, TextWorld may be key to creating a curriculum of learning tasks and helping agents scale to human-made games.

Ammanabrolu and Riedl (2019a) present the Knowledge Graph DQN or KG-DQN, an approach where a knowledge graph built during exploration is used as a state representation for a deep reinforcement learning based agent. They also use question-answering techniques—asking the question of what action best next to take—to pretrain a deep Q-network. These techniques are then shown to aid in overcoming the twin challenges of a partially observable state space and a combinatorial action space. Ammanabrolu and Riedl (2019b) further expand on this work, exploring methods of transferring control policies in text-games, using knowledge graphs to seed an agent with useful commonsense knowledge and transfer knowledge between different games within a domain. They show that training on a source game and transferring to target game within the same genre—e.g. horror or slice of life—is more effective and efficient than training from scratch on the target game.

Finally, although not a sequential decision making problem, Light (Urbanek et al. 2019) is a crowdsourced dataset of text-adventure game dialogues. The authors demonstrate that supervised training of transformer-based models have the ability to generate contextually relevant dialog, actions, and emotes.

4 Jericho Environment

Jericho is an open-source[1] Python-based IF environment, which provides an OpenAI-Gym-like interface (Brockman et al. 2016) for learning agents to connect with IF games.

4.1 Template-Based Action Generation

Template-based action generation involves first selecting a template, then choosing words to fill in the blanks in that template. The set of game-specific templates are identified by decompiling a game to extract the possible subroutines – each template corresponds to a different subroutine. Templates contain a maximum of two blanks to be filled. Additionally, game-specific vocabulary is also extracted and provides a list of all words recognized by the game's parser. Combining templates with vocabulary yields a game-specific action space, which is far more tractable than operating in the pure vocabulary space.

4.2 World Object Tree

The world object tree is a semi-interpretable representation of game state that IF games use internally to codify the relationship between the objects and locations that populate the game world. Each object in the tree has a *parent*, *child*, and *sibling*. These relationships between objects are used to encode presence: a location contains children objects that correspond to the items present at that location. Similarly, there is an object corresponding to the player, whose parent is the player's location and whose children are the objects in the player's inventory.

Jericho's ability to extract world-object-trees forms the basis for world-change-detection (described in the next subsection) and ground-truth object detection. Ground truth object detection searches the object tree for all non-player objects present at the current location, thus sidestepping the challenge of identifying interactive objects from a location's description.

4.3 World Change Detection

Jericho has the ability to best-guess detect whether an action changed the world state of the game. Using this facility, it's possible to identify the valid actions from a state by performing a search over template-based actions and excluding any actions that don't change the world state. We demonstrate the feasibility of valid action detection by training a choice-based learner, DRRN (Section 5.2). World change detection is based on identifying changes to the world-object tree[2] and can fail to detect valid actions whose effects alter only global variables instead of the object tree. In practice, these failures are rare.

4.4 Supported Games

For supported games, Jericho is able to detect game score, move count, and world change. Jericho supports a set of fifty-six human-made IF games that cover a variety of genres: dungeon crawl, Sci-Fi, mystery, comedy, and horror. Games were selected from classic Infocom titles such as *Zork* and *Hitchhiker's Guide to the Galaxy*, as well as newer, community-created titles like *Anchorhead* and *Afflicted*. All supported games use a point-based scoring system, which serves as the agent's reward.

Unsupported games may be played through Jericho, without the support of score detection, move counts, or world-change detection. There exists a large collection of over a thousand unsupported games [3], which may be useful for unsupervised pretraining or intrinsic motivation.

[1] Jericho is available at https://github.com/microsoft/jericho.

[2] Game trees are standardized representations for all Z-Machine games. To learn more see https://inform-fiction.org/zmachine/standards/z1point1/index.html

[3] https://github.com/BYU-PCCL/z-machine-games

4.5 Identifying Valid Actions

Valid actions are actions recognized by the game's parser that cause changes in the game state. When playing new games, identifying valid actions is one of the primary difficulties encountered by humans and agents alike. Jericho identifies valid actions using the following procedure:

Algorithm 1 Procedure for Identifying Valid Actions

$\mathcal{E} \leftarrow$ Jericho environment
$\mathcal{T} \leftarrow$ Set of action templates
$o \leftarrow$ Textual observation
$\mathcal{P} \leftarrow \{p_1 \dots p_n\}$ Interactive objects identified with noun-phrase extraction or world object tree.
$Y \leftarrow \varnothing$ List of valid actions
$s \leftarrow \mathcal{E}.save()$ – Save current game state
for template $u \in \mathcal{T}$ **do**
    for all combinations $p_1, p_2 \in \mathcal{P}$ **do**
        Action $a \leftarrow u \Leftarrow p_1, p_2$
        if $\mathcal{E}.world\_changed(\mathcal{E}.step(a))$ **then**
            $Y \leftarrow Y \cup a$
            $\mathcal{E}.load(s)$ – Restore saved game state
return Y

4.6 Handicaps

To ease the difficulty of IF games, Jericho has the option of the following handicaps:

- *Inputs*: Addition of a location description, player's inventory, and game score.
- *Outputs*: Game-specific templates $\mathcal{T}$ and vocabulary $\mathcal{V}$.
- *World Objects*: Use of world object tree for identifying interactive objects or player's current location.
- *Valid Actions*: Use of world-changed-detection to identify valid actions.

For reproducibilty, we report the handicaps used by all algorithms in the next section and encourage future work to do the same.

5 Algorithms

IF game playing has been approached from the perspective single-game agents that are trained and evaluated on the same game and general game playing agents which are trained and evaluated on different sets of games. In this section we present three agents: a choice-based single-game agent (DRRN), a parser-based single-game agent (TDQN), and a parser-based general-game agent (NAIL).

5.1 Common Input Representation

The input encoder ϕ_o converts observations into vectors using the following process: Text observations are tokenized by a SentencePiece model (Kudo and Richardson 2018) using an 8000-large vocab trained on strings extracted from http://www.allthingsjacq.com/index.html sessions of humans playing a variety of different IF games. Tokenized observations are processed by separate GRU encoders for the narrative (i.e., the game's response to last action), description of current location, contents of inventory, and previous text command. The outputs of these encoders are concatenated into a vector ν_o. DRRN and Template-DQN build on this common input representation.

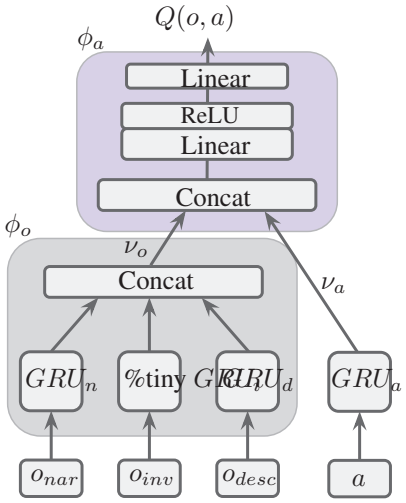

Figure 2: DRRN architecture estimates a joint Q-Value $Q(o, a)$ over the observation o and an action a. The observation encoder ϕ_o uses separate GRUs to process the narrative text o_{nar}, the players inventory o_{inv}, and the location description o_{desc} into a vector ν_o. DRRN uses a separate GRU_a for processing action text into a vector ν_a. The action-scorer ϕ_a concatenates the input and action vectors and outputs a scalar Q-Value.

5.2 DRRN

The Deep Reinforcement Relevance Network (DRRN) (He et al. 2016) is an algorithm for *choice-based* games that present a set of valid actions $A_{valid}(s)$ at every game state s. We re-implement DRRN using a GRU $\phi_{act}(a)$ to encode each valid action into a vector ν_a, which is concatenated with the encoded observation vector ν_o. Using this combined vector, DRRN then computes a Q-Value $Q(o, a)$ estimating the total discounted reward expected if action a is taken and π_{DRRN} is followed thereafter. This procedure is repeated for each valid action $a_i \in A_{valid}(s)$. Action selection is performed by sampling from a softmax distribution over $Q(o, a_i)$. The network is updated by sampling a minibatch of transitions $(o, a, r, o', A_{valid}(s')) \sim \mathcal{D}$ from the prioritized replay memory (Schaul et al. 2016) and minimizing the temporal difference error $\delta = r + \gamma * \max'_a Q(o', a') - Q(o, a)$. From an optimization perspective, rather than performing a separate forward pass for each valid action, we batch valid-actions and perform a single forward pass computing Q-Values for all valid actions.

In order to make DRRN applicable to parser-based IF games, it's necessary to identify the list of valid actions available at each step. This is accomplished through a search over template-based actions, pruned by Jericho's world

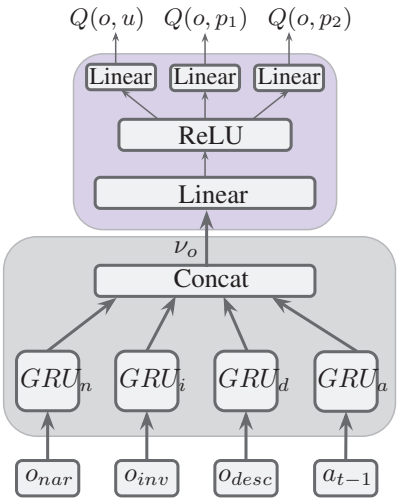

Figure 3: Template-DQN estimates Q-Values $Q(o,u)$ for all templates $u \in \mathcal{T}$ and $Q(o,p)$ for all vocabulary $p \in \mathcal{V}$. Similar to DRRN, separate GRUs are used to encode each component of the observation, including the text of the previous action a_{t-1}.

change detection (Algorithm 1). This handicap bypasses language generation, one of the challenges of IF games.

5.3 Template-DQN

LSTM-DQN (Narasimhan, Kulkarni, and Barzilay 2015) is an agent for *parser-based* games that handles the combinatorial action space by generating *verb-object* actions using a set possible verbs and possible objects. Specifically, LSTM-DQN uses two output layers to estimate Q-Values over possible verbs and objects. Actions are selected by pairing the maximally valued verb with the maximally valued noun.

Template-DQN (TDQN) extends LSTM-DQN by incorporating template-based action generation (Section 4.1). This is accomplished using three output heads: one for estimating Q-Values over templates $Q(o,u); u \in \mathcal{T}$ and two for estimating Q-Values $Q(o,p_1), Q(o,p_2); p \in \mathcal{V}$ to fill in the blanks of the template using vocabulary.

The largest action space considered in the original LSTM-DQN paper was 222 actions (6 verbs and 22 objects). In contrast, *Zork1* has 237 templates with a 697 word vocabulary, yielding an action space of 115 million. Computationally, this space is too large to naively explore as the vast majority of actions will be un-grammatical or contextually irrelevant. To help guide the agent towards valid actions, we introduce a supervised binary-cross entropy loss based on the valid actions in the current state. Valid actions are identified using the same procedure as in DRRN. This loss is mixed with the standard temporal difference error during each update.

We further optimize by decoding large batches of actions with each forward pass over the network and executing them sequentially until a valid action is found. When decoding actions that are often invalid, this provides a considerable speedup compared to performing an separate forward pass for each action.

5.4 NAIL

NAIL (Hausknecht et al. 2019) is the state-of-the-art agent for general interactive fiction game playing (Atkinson et al. 2018). Rather than being trained and evaluated on a single game, NAIL is designed to play unseen IF games and accumulate as much score as possible in a single episode. Operating without the handicaps outlined in Section 4.6, NAIL employs a set of manually-created heuristics to build a map of objects and locations, reason about which actions were valid or invalid, and uses a web-based language model to decide how to interact with novel objects. We include NAIL's performance on the same set of games in order to highlight the difference between general and single-game playing agents, and to provide a reference scores for future work in general IF game playing.

6 Experiments

We evaluate the agents across a set of thirty-two Jericho-supported games with the aims of 1) showing the feasibility of reinforcement learning on a variety of different IF games, 2) creating a reproducible benchmark for future work, 3) investigating the difference between choice-based and template-based action spaces, and 4) comparing performance of general IF game playing agents (NAIL), single-game agents (DRRN and TDQN), and a random agent (RAND) which uniformly sample commands from a set of canonical actions: {*north, south, east, west, up, down, look, inventory, take all, drop, yes*}.

Results in Table 1, supported by learning curves in Figure 4 show that reinforcement learning is a viable approach for optimizing IF game playing agents across many different types of games. Experimentally, TDQN and DRRN agents are trained and evaluated on each game individually, but their hyperparameters are fixed across the different games.

In order to quantify overall progress towards story completion, we normalize agent score by maximum possible game score averaged across all games. The resulting progress scores are as follows: **RANDOM** 1.8%**, NAIL** 4.9%**, TDQN** 6.1%**, and DRRN** 1.7% **completion**.

Comparing the different agents, the random agent shows that more than simple navigation and take actions are needed to succeed at the vast majority of games. Comparing DRRN to TDQN highlights the utility of choice-based game playing agents who need only estimate Q-Values over pre-identified valid-actions. In contrast, TDQN needs to estimate Q-Values over the full space of templates and vocabulary words. As a result, we observed that TDQN was more prone to overestimating Q-Values due to the Q-Learning update computing a max over a much larger number of possible actions.

Comparing the general game playing NAIL agent to single-game agents: the NAIL agent performs surprisingly well considering it uses no handicaps, no training period, and plays the game for only a single episode. It should be noted that the NAIL agent was developed on many of the same games used in this evaluation. The fact that the reinforcement learning agents outperform NAIL serves to highlight the difficulty of engineering an IF agent as well as

the promise of learning policies from data rather than hand-coding heuristics.

Broadly, all algorithms have a long way to go before they are solving games of even average difficulty. Five games prove too challenging for any agents to get a nonzero reward. Games like *Anchorhead* are highly complex and others pose difficult exploration problems like *9:05* which features only a single terminal reward indicating success or failure at the end of the episode.

Additional experiment details and hyperparameters are located in the supplementary material.

Game	$\|\mathcal{T}\|$	$\|\mathcal{V}\|$	RAND	NAIL	TDQN	DRRN
905	82	296	0	0	0	0
acorncourt	151	343	0	0	.05	**.33**
anchor	260	2257	0	0	0	0
advent	189	786	.1	.1	.1	.1
adventureland	156	398	0	0	0	**.21**
afflicted	146	762	0	0	.02	**.03**
awaken	159	505	0	0	0	0
balances	156	452	0	**.2**	.09	**.2**
deephome	173	760	0	**.04**	0	0
detective	197	344	.32	.38	.47	**.55**
dragon	177	1049	0	**.02**	-.21	-.14
enchanter	290	722	0	0	.02	**.05**
gold	200	728	0	.03	**.04**	0
inhumane	141	409	0	0	0	0
jewel	161	657	0	**.02**	0	**.02**
karn	178	615	0	**.01**	0	**.01**
library	173	510	0	.03	.21	**.57**
ludicorp	187	503	**.09**	.06	.04	**.09**
moonlit	166	669	0	0	0	0
omniquest	207	460	0	.11	**.34**	.1
pentari	155	472	0	0	.25	**.39**
reverb	183	526	0	0	.01	**.16**
snacktime	201	468	0	0	**.19**	0
sorcerer	288	1013	.01	.01	.01	**.05**
spellbrkr	333	844	.04	**.07**	.03	.06
spirit	169	1112	**.01**	0	0	0
temple	175	622	0	.21	**.23**	.21
tryst205	197	871	0	.01	0	**.03**
yomomma	141	619	0	0	0	**.01**
zenon	149	401	0	0	0	0
zork1	237	697	0	.03	.03	**.09**
zork3	214	564	.03	**.26**	0	.07
ztuu	186	607	0	0	.05	**.22**

Table 1: **Normalized scores** for Jericho-supported games. Results are averaged over the last hundred episodes of training and across five independent training runs (i.e., with different seeds for initializing the environment and agent sampling process) conducted for each algorithm.

7 Notable Games

Jericho supports a vast array of games, covering a diverse set of structures and genres. These games provide us with different challenges from the perspective of reinforcement learning based agents. In this section, we highlight some specific challenges posed by Jericho supported games and provide notable examples for each of the types of challenges in addition to examples of games that the two types of deep reinforcement learning agents do well on. Learning curves for some of these examples using DRRN and TDQN are presented (Figure 4), underscoring the difficulties current reinforcement learning agents have in solving these games and showcasing effective training paradigms for different games.

7.1 Sanity Checks

The first set of games are essentially sanity checks, i.e. they are games that can be solved relatively well by existing agents. These games thus fall on the lower end on the difficulty spectrum and serve as good initial testbeds for developing new agents.

Detective in particular is one of the easier games, and with existing agents such as the random agent and NAIL being able to solve the majority of the game. The relatively good performance of the random agent is likely due to the game mostly requiring only navigational actions in order to accumulate score. On this game, TDQN has comparable performance to DRRN. *Acorncourt* also serves as a sanity check, albeit a more difficult one—with the DRRN outperforming all other agents. This game requires a higher proportion of higher complexity actions, which make generation—such as in the case of TDQN—more difficult. *Omniquest* is an example of a dungeon-crawler style game where TDQN outperforms the rest of the agents. In this game, due to there being a relatively smaller number of valid templates as compared to valid actions—i.e. many valid actions come from the same template—the TDQN has an effective search space that is smaller than the DRRN.

7.2 Varying Rewards

Most IF games provide you with positive scores in varying increments as you achieve objectives and negative scores for failing them. This reward structure is similar to most games in general and gives the player an indication of relative progress within the game. Some games such as *Deephome* and *Adventure*, however, provide relatively unintuitive scoring functions that can prove to be a challenge for reinforcement learning agents.

Deephome gives you an additional point of score for each new room that you visit in addition to rewards for achieving game objectives. This encourages exploration but could also prove tricky for an agent as it may not be necessary to finish the game. In *Adventure*, you start with a score of 36, and as the game progresses you are first given mostly negative scores followed by positive scores until you finish the game. As seen in Figure 4, this gives a reinforcement learning agent no indication that it is progressing in the short term.

7.3 Moonshots

Here we highlight some of the most difficult games in Jericho, current agents are quite far from being able to solve them. *Zork1* is one of the original IF games and heavily influences later games using this medium. The game can best be described as a dungeon-crawler in which a player must make their way through a large dungeon, collecting treasures and fighting monsters along the way. The collection of these treasures forms the basis of *Zork1's* scoring system, although some score is rewarded at intermediate steps to aid in finding the treasures. Being a dungeon-crawler, *Zork1* features branching game path in terms of reward collection as well as stochasticity. The game can be completed

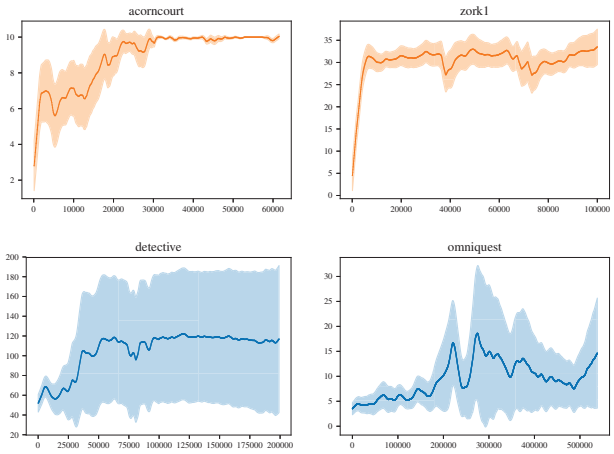

Figure 4: Episode score as a function of training steps for DRRN (top) and TDQN (bottom). Shaded regions denote standard deviation across five independent runs for each game. Additional learning curves in supplementary material.

in many different ways, often affected by the random movement of a thief and number of hits required to kill monsters. It is also interesting to note that NAIL and TDQN perform comparably on *Zork1*, with DRRN far outperforming them—indicating the difficulty of language generation in such a large state space. It has also been the subject of much prior work on IF game-playing agents (Zahavy et al. 2018; Yin and May 2019).

Anchorhead is a Lovecraftian horror game where the player must wade through a complex puzzle-based narrative. The game features very long term dependencies in piecing together the information required to solve its puzzles and is complex enough that it has been the subject of prior work on cognition in script learning and drama management (Giannatos et al. 2011). This complexity is further reflected in the size of the vocab and number of templates—it has the largest action space of any Jericho supported game. None of our agents are able to accumulate any score on this game.

Although *Anchorhead's* game structure is more sequential than *Zork1*, it also contains a more sparse reward—often giving you a positive score only after the completion of a puzzle. It is also stochastic, with the exact solution depending on the random seed supplied when the game is started.

8 Future Work

Unsupervised learning: DRRN and TDQN agents were trained and evaluated on individual games. While this is sufficient for a proof-of-concept, it falls short of demonstrating truly general IF game playing. To this end, it's valuable to evaluate agents on a separate set of games which they have not been trained on. In the Jericho framework, we propose to use the set of Jericho supported games as a *test set* and the larger set of unsupported games[4] as the *training set*. In this paradigm, it's necessary have a strong unsupervised learning component to guide the agent's exploration and learning since unsupported games do not provide rewards, and in fact many IF games do not have scores. We hypothesize that *surrogate reward functions*, like novelty-based rewards (Bellemare et al. 2016; Pathak et al. 2017), will be useful for discovering locations, successful interactions and objects.

Better Template-based Agents: There are several directions for creating better template-based agents by improving on the limitations of TDQN: When generating actions, TDQN assumes independence between templates and vocabulary words. To understand the problem with this assumption consider the templates "go _" and "take _" and the vocabulary words "north, apple". Independently, "take" and "north" may have the highest Q-Values together yield the invalid action "take north". Conditional generation of words based on the chosen template may go far to improve the quality of TDQN's actions.

Second, recent work on transformer-based neural architectures has yielded impressive gains in many NLP tasks (Devlin et al. 2018), including text-adventure game dialogues (Urbanek et al. 2019). We expect these advances may be applicable to human-made IF games, but will need to be adapted from a supervised training regime into reinforcement learning.

9 Conclusion

Interactive Fiction games are rich narrative adventures that challenge even skilled human players. In contrast to other video game environments like ALE (Machado et al. 2017), Vizdoom (Kempka et al. 2016), and Malmo (Johnson et al. 2016), IF games stress natural language understanding and commonsense reasoning, and feature combinatorial action spaces. To aid in the study of these environment, we introduce Jericho, an experimental platform with the key of feature of extracting game-specific action templates and vocabulary. Using these features, we proposed a novel template-based action space which serves to reduce the complexity of full scale language generation. Using this space, we introduced the Template-DQN agent, which generates actions first by selecting a template then filling in the blanks with words from the vocabulary.

We evaluated Template-DQN as well as a choice-based agent DRRN and a general IF agent NAIL on a set of thirty-two human-made IF games. Overall, DRRN outperformed the other agents with Template-DQN in second place. However, in many senses these agents represent very different training paradigms, sets of assumptions, and levels of handicap. Rather than comparing agents we aim to provide benchmark results for future work in these three categories of IF game playing. All in all, we believe Jericho can help the community propel research on language understanding agents and expect these environments can serve the community as benchmarks for years to come.

References

Ammanabrolu, P., and Riedl, M. O. 2019a. Playing text-adventure games with graph-based deep reinforcement

[4]https://github.com/BYU-PCCL/z-machine-games

learning. In *Proceedings of 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*.

Ammanabrolu, P., and Riedl, M. O. 2019b. Transfer in deep reinforcement learning using knowledge graphs. *CoRR* abs/1908.06556.

Atkinson, T.; Baier, H.; Copplestone, T.; Devlin, S.; and Swan, J. 2018. The text-based adventure AI competition.

Bellemare, M. G.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; and Munos, R. 2016. Unifying count-based exploration and intrinsic motivation. *CoRR* abs/1606.01868.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym.

Côté, M.-A.; Kádár, A.; Yuan, X.; Kybartas, B.; Barnes, T.; Fine, E.; Moore, J.; Hausknecht, M.; Asri, L. E.; Adada, M.; Tay, W.; and Trischler, A. 2018. Textworld: A learning environment for text-based games. *CoRR* abs/1806.11532.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR* abs/1810.04805.

Fulda, N.; Ricks, D.; Murdoch, B.; and Wingate, D. 2017. What can you do with a rock? affordance extraction via word embeddings. In *IJCAI*, 1039–1045.

Giannatos, S.; Nelson, M. J.; Cheong, Y.-G.; and Yannakakis, G. N. 2011. Suggesting New Plot Elements for an Interactive Story. In *In Workshop on Intellignet Narrative Technologies (INT'11)*.

Gibson, J. J. 1977. *"The theory of affordances," in Perceiving, Acting, and Knowing. Towards an Ecological Psychology.* Number eds Shaw R., Bransford J. Hoboken, NJ: John Wiley & Sons Inc.

Hausknecht, M. J.; Loynd, R.; Yang, G.; Swaminathan, A.; and Williams, J. D. 2019. NAIL: A general interactive fiction agent. *CoRR* abs/1902.04259.

He, J.; Chen, J.; He, X.; Gao, J.; Li, L.; Deng, L.; and Ostendorf, M. 2016. Deep reinforcement learning with a natural language action space. In *ACL*.

Jaderberg, M.; Mnih, V.; Czarnecki, W. M.; Schaul, T.; Leibo, J. Z.; Silver, D.; and Kavukcuoglu, K. 2016. Reinforcement learning with unsupervised auxiliary tasks. *CoRR* abs/1611.05397.

Johnson, M.; Hofmann, K.; Hutton, T.; and Bignell, D. 2016. The malmo platform for artificial intelligence experimentation. In *IJCAI*, IJCAI'16, 4246–4247. AAAI Press.

Kempka, M.; Wydmuch, M.; Runc, G.; Toczek, J.; and Jaśkowski, W. 2016. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *CIG*, 341–348. Santorini, Greece: IEEE. The best paper award.

Kudo, T., and Richardson, J. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR* abs/1808.06226.

Machado, M. C.; Bellemare, M. G.; Talvitie, E.; Veness, J.; Hausknecht, M. J.; and Bowling, M. 2017. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *CoRR* abs/1709.06009.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Narasimhan, K.; Kulkarni, T. D.; and Barzilay, R. 2015. Language understanding for text-based games using deep reinforcement learning. In *EMNLP*, 1–11.

OpenAI; Andrychowicz, M.; Baker, B.; Chociej, M.; Józefowicz, R.; McGrew, B.; Pachocki, J. W.; Pachocki, J.; Petron, A.; Plappert, M.; Powell, G.; Ray, A.; Schneider, J.; Sidor, S.; Tobin, J.; Welinder, P.; Weng, L.; and Zaremba, W. 2018. Learning dexterous in-hand manipulation. *CoRR* abs/1808.00177.

Pathak, D.; Agrawal, P.; Efros, A. A.; and Darrell, T. 2017. Curiosity-driven exploration by self-supervised prediction. *CoRR* abs/1705.05363.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2016. Prioritized experience replay. In *International Conference on Learning Representations*.

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

Thrun, S.; Burgard, W.; and Fox, D. 2005. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press.

Urbanek, J.; Fan, A.; Karamcheti, S.; Jain, S.; Humeau, S.; Dinan, E.; Rocktäschel, T.; Kiela, D.; Szlam, A.; and Weston, J. 2019. Learning to speak and act in a fantasy text adventure game. volume abs/1903.03094.

Yin, X., and May, J. 2019. Comprehensible context-driven text game playing. *CoRR* abs/1905.02265.

Yuan, X.; Côté, M.; Sordoni, A.; Laroche, R.; des Combes, R. T.; Hausknecht, M. J.; and Trischler, A. 2018. Counting to explore and generalize in text-based games. *CoRR* abs/1806.11525.

Yuan, X.; Côté, M.-A.; Fu, J.; Lin, Z.; Pal, C.; Bengio, Y.; and Trischler, A. 2019. Interactive language learning by question answering.

Zahavy, T.; Haroush, M.; Merlis, N.; Mankowitz, D. J.; and Mannor, S. 2018. Learn what not to learn: Action elimination with deep reinforcement learning. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc. 3562–3573.

Zelinka, M. 2018. Using reinforcement learning to learn how to play text-based games. *CoRR* abs/1801.01999.