Transfer in Deep Reinforcement Learning using Knowledge Graphs

Prithviraj Ammanabrolu
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA
`raj.ammanabrolu@gatech.edu`

Mark O. Riedl
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA
`riedl@cc.gatech.edu`

Abstract

Text adventure games, in which players must make sense of the world through text descriptions and declare actions through text descriptions, provide a stepping stone toward grounding action in language. Prior work has demonstrated that using a knowledge graph as a state representation and question-answering to pre-train a deep Q-network facilitates faster control policy learning. In this paper, we explore the use of knowledge graphs as a representation for domain knowledge transfer for training text-adventure playing reinforcement learning agents. Our methods are tested across multiple computer generated and human authored games, varying in domain and complexity, and demonstrate that our transfer learning methods let us learn a higher-quality control policy faster.

1 Introduction

Text adventure games, in which players must make sense of the world through text descriptions and declare actions through natural language, can provide a stepping stone toward more real-world environments where agents must communicate to understand the state of the world and affect change in the world. Despite the steadily increasing body of research on text-adventure games (Bordes et al., 2010; He et al., 2016; Narasimhan et al., 2015; Fulda et al., 2017; Haroush et al., 2018; Côté et al., 2018; Tao et al., 2018; Ammanabrolu and Riedl, 2019), and in addition to the ubiquity of deep reinforcement learning applications (Parisotto et al., 2016; Zambaldi et al., 2019), teaching an agent to play text-adventure games remains a challenging task. Learning a control policy for a text-adventure game requires a significant amount of exploration, resulting in training runs that take hundreds of thousands of simulations (Narasimhan et al., 2015; Ammanabrolu and Riedl, 2019).

One reason that text-adventure games require so much exploration is that most deep reinforcement learning algorithms are trained on a task without a real prior. In essence, the agent must learn everything about the game from only its interactions with the environment. Yet, text-adventure games make ample use of commonsense knowledge (e.g., an axe can be used to cut wood) and genre themes (e.g., in a horror or fantasy game, a coffin is likely to contain a vampire or other undead monster). This is in addition to the challenges innate to the text-adventure game itself—games are puzzles—which results in inefficient training.

Ammanabrolu and Riedl (2019) developed a reinforcement learning agent that modeled the text environment as a knowledge graph and achieved state-of-the-art results on simple text-adventure games provided by the TextWorld (Côté et al., 2018) environment. They observed that a simple form of transfer from very similar games greatly improved policy training time. However, games beyond the toy TextWorld environments are beyond the reach of state-of-the-art techniques.

In this paper, we explore the use of knowledge graphs and associated neural embeddings as a medium for domain transfer to improve training effectiveness on new text-adventure games. Specifically, we explore transfer learning at multiple levels and across different dimensions. We first look at the effects of playing a text-adventure game given a strong prior in the form of a knowledge graph extracted from generalized textual walk-throughs of interactive fiction as well as those made specifically for a given game. Next, we explore the transfer of control policies in deep Q-learning (DQN) by pre-training portions of a deep Q-network using question-answering and by DQN-to-DQN parameter transfer between games. We evaluate these techniques on two different sets of human authored and computer generated

games, demonstrating that our transfer learning methods enable us to learn a higher-quality control policy faster.

2 Background and Related Work

Text-adventure games, in which an agent must interact with the world entirely through natural language, provide us with two challenges that have proven difficult for deep reinforcement learning to solve (Narasimhan et al., 2015; Haroush et al., 2018; Ammanabrolu and Riedl, 2019): (1) The agent must act based only on potentially incomplete textual descriptions of the world around it. The world is thus partially observable, as the agent does not have access to the state of the world at any stage. (2) the action space is combinatorially large—a consequence of the agent having to declare commands in natural language. These two problems together have kept commercial text adventure games out of the reach of existing deep reinforcement learning methods, especially given the fact that most of these methods attempt to train on a particular game from scratch.

Text-adventure games can be treated as partially observable Markov decision processes (POMDPs). This can be represented as a 7-tuple of $\langle S, T, A, \Omega, O, R, \gamma \rangle$: the set of environment states, conditional transition probabilities between states, words used to compose text commands, observations, conditional observation probabilities, the reward function, and the discount factor respectively (Côté et al., 2018).

Multiple recent works have explored the challenges associated with these games (Bordes et al., 2010; He et al., 2016; Narasimhan et al., 2015; Fulda et al., 2017; Haroush et al., 2018; Côté et al., 2018; Tao et al., 2018; Ammanabrolu and Riedl, 2019). Narasimhan et al. (2015) introduce the LSTM-DQN, which learns to score the action verbs and corresponding objects separately and then combine them into a single action. He et al. (2016) propose the Deep Reinforcement Relevance Network that consists of separate networks to encode state and action information, with a final Q-value for a state-action pair that is computed between a pairwise interaction function between these. Haroush et al. (2018) present the Action Elimination Network (AEN), which restricts actions in a state to the top-k most likely ones, using the emulator's feedback. Hausknecht et al. (2019b) design an agent that uses multiple mod-

ules to identify a general set of game play rules for text games across various domains. None of these works study how to transfer policies between different text-adventure games in any depth and so there exists a gap between the two bodies of work.

Transferring policies across different text-adventure games requires implicitly learning a mapping between the games' state and action spaces. The more different the domain of the two games, the harder this task becomes. Previous work (Ammanabrolu and Riedl, 2019) introduced the use of knowledge graphs and question-answering pre-training to aid in the problems of partial observability and a combinatorial action space. This work made use of a system called TextWorld (Côté et al., 2018) that uses grammars to generate a series of similar (but not exact same) games. An oracle was used to play perfect games and the traces were used to pre-train portions of the agent's network responsible for encoding the observations, graph, and actions. Their results show that this form of pre-training improves the quality of the policy at convergence it does not show a significant improvement in the training time required to reach convergence. Further, it is generally unrealistic to have a corpus of very similar games to draw from. We build on this work, and explore modifications of this algorithm that would enable more efficient transfer in text-adventure games.

Work in transfer in reinforcement learning has explored the idea of transferring skills (Konidaris and Barto, 2007; Konidaris et al., 2012) or transferring value functions/policies (Liu and Stone, 2006). Other approaches attempt transfer in model-based reinforcement learning (Taylor et al., 2008; Nguyen et al., 2012; Gasic et al., 2013; Wang et al., 2015; Joshi and Chowdhary, 2018), though traditional approaches here rely heavily on hand crafting state-action mappings across domains. Narasimhan et al. (2017) learn to play games by predicting mappings across domains using a both deep Q-networks and value iteration networks, finding that that grounding the game state using natural language descriptions of the game itself aids significantly in transferring useful knowledge between domains.

In transfer for deep reinforcement learning, Parisotto et al. (2016) propose the Actor-Mimic network which learns from expert policies for a source task using policy distillation and then ini-

tializes the network for a target task using these parameters. Yin and Pan (2017) also use policy distillation, using task specific features as inputs to a multi-task policy network and use a hierarchical experience sampling method to train this multi-task network. Similarly, Rusu et al. (2016) attempt to transfer parameters by using frozen parameters trained on source tasks to help learn a new set of parameters on target tasks. Rajendran et al. (2017) attempt something similar but use attention networks to transfer expert policies between tasks. These works, however, do not study the requirements for enabling efficient transfer for tasks rooted in natural language, nor do they explore the use of knowledge graphs as a state representation.

3 Knowledge Graphs for DQNs

A knowledge graph is a directed graph formed by a set of semantic, or RDF, triples in the form of $\langle subject, relation, object \rangle$—for example, $\langle vampires, are, undead \rangle$. We follow the open-world assumption that what is not in our knowledge graph can either be true or false.

Ammanabrolu and Riedl (2019) introduced the Knowledge Graph DQN (KG-DQN) and touched on some aspects of transfer learning, showing that pre-training portions of the deep Q-network using question answering system on perfect playthroughs of a game increases the quality of the learned control policy for a generated text-adventure game. We build on this work and use KG-DQN to explore transfer with both knowledge graphs and network parameters. Specifically we seek to transfer skills and knowledge from (a) static text documents describing game play and (b) from playing one text-adventure game to a second complete game in in the same genre (e.g., horror games). The rest of this section describes KG-DQN in detail and summarizes our modifications.[1]

For each step that the agent takes, it automatically extracts a set of RDF triples from the received observation through the use of OpenIE (Angeli et al., 2015) in addition to a few rules to account for the regularities of text-adventure games. The graph itself is more or less a map of the world, with information about objects' affordances and attributes linked to the rooms that they are place in in a map. The graph also makes a distinction with respect to items that are in the agent's

[1]We use the implementation of KG-DQN found at https://github.com/rajammanabrolu/KG-DQN
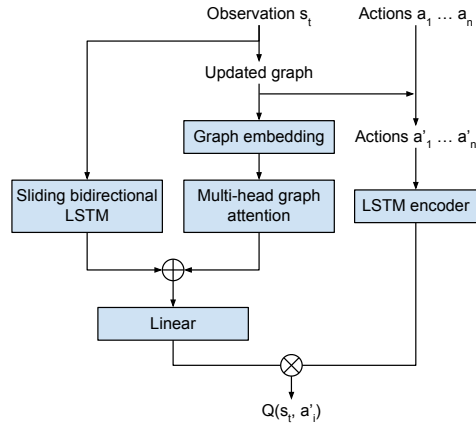
Figure 1: The KG-DQN architecture.

possession or in their immediate surrounding environment. We make minor modifications to the rules used in Ammanabrolu and Riedl (2019) to better achieve such a graph in general interactive fiction environments.

The agent also has access to all actions accepted by the game's parser, following Narasimhan et al. (2015). For general interactive fiction environments, we develop our own method to extract this information. This is done by extracting a set of templates accepted by the parser, with the objects or noun phrases in the actions replaces with a OBJ tag. An example of such a template is "place OBJ in OBJ". These OBJ tags are then filled in by looking at all possible objects in the given vocabulary for the game. This action space is of the order of $A = \mathcal{O}(|V| \times |O|^2)$ where V is the number of action verbs, and O is the number of distinct objects in the world that the agent can interact with. As this is too large a space for a RL agent to effectively explore, the knowledge graph is used to prune this space by ranking actions based on their presence in the current knowledge graph and the relations between the objects in the graph as in Ammanabrolu and Riedl (2019).

The architecture for the deep Q-network consists of two separate neural networks—encoding state and action separately—with the final Q-value for a state-action pair being the result of a pairwise interaction function between the two (Figure 1). We train with a standard DQN training loop; the policy is determined by the Q-value of a particular state-action pair, which is updated using

the Bellman equation (Sutton and Barto, 2018):

$$Q_{t+1}(s_{t+1}, a_{t+1}) = \\ E[r_{t+1} + \gamma \max_{a \in A_t} Q_t(s, a) | s_t, a_t] \quad (1)$$

where γ refers to the discount factor and r_{t+1} is the observed reward. The whole system is trained using prioritized experience replay Lin (1993), a modified version of ϵ-greedy learning, and a temporal difference loss that is computed as:

$$L(\theta) = r_{k+1} + \\ \gamma \max_{\mathbf{a} \in \mathbf{A_{k+1}}} Q(\mathbf{s_t}, \mathbf{a}; \theta) - Q(\mathbf{s_t}, \mathbf{a_t}; \theta) \quad (2)$$

where $\mathbf{A_{k+1}}$ represents the action set at step $k + 1$ and $\mathbf{s_t}, \mathbf{a_t}$ refer to the encoded state and action representations respectively.

4 Knowledge Graph Seeding

In this section we consider the problem of transferring a knowledge graph from a static text resource to a DQN—which we refer to as *seeding*. KG-DQN uses a knowledge graph as a state representation and also to prune the action space. This graph is built up over time, through the course of the agent's exploration. When the agent first starts the game, however, this graph is empty and does not help much in the action pruning process. The agent thus wastes a large number of steps near the beginning of each game exploring ineffectively.

The intuition behind seeding the knowledge graph from another source is to give the agent a prior on which actions have a higher utility and thereby enabling more effective exploration. Text-adventure games typically belong to a particular genre of storytelling—e.g., horror, sci-fi, or soap opera—and an agent is at a distinct disadvantage if it doesn't have any genre knowledge. Thus, the goal of seeding is to give the agent a strong prior.

This seed knowledge graph is extracted from online general text-adventure guides as well as game/genre specific guides when available.[2] The graph is extracted from this the guide using a subset of the rules described in Section 3 used to extract information from the game observations, with the remainder of the RDF triples coming from OpenIE. There is no map of rooms in the environment that can be built, but it is possible to

---

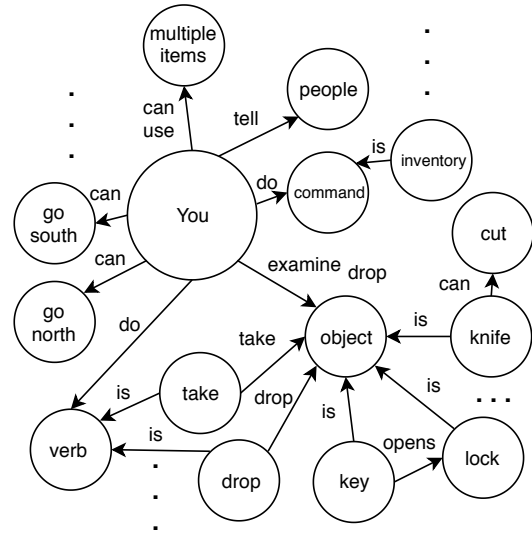[2]An example of a guide we use is found here http://www.microheaven.com/IFGuide/step3.html

Figure 2: Select partial example of what a seed knowledge graph looks like. Ellipses indicate other similar entities and relations not shown.

extract information regarding affordances of frequently occurring objects as well as common actions that can be performed across a wide range of text-adventure games. This extracted graph is thus potentially disjoint, containing only this generalizable information, in contrast to the graph extracted during the rest of the exploration process. An example of a graph used to seed KG-DQN is given in Fig. 2. The KG-DQN is initialized with this knowledge graph.

5 Task Specific Transfer

The overarching goal of transfer learning in text-adventure games is to be able to train an agent on one game and use this training on to improve the learning capabilities of another. There is growing body of work on improving training times on target tasks by transferring network parameters trained on source tasks (Rusu et al., 2016; Yin and Pan, 2017; Rajendran et al., 2017). Of particular note is the work by Rusu et al. (2016), where they train a policy on a source task and then use this to help learn a new set of parameters on a target task. In this approach, decisions made during the training of the target task are jointly made using the frozen parameters of the transferred policy network as well as the current policy network.

Our system first trains a question-answering system (Chen et al., 2017) using traces given by an oracle, as in Section 4. For commercial text-adventure games, these traces take the form of state-action pairs generated using perfect walk-

4

through descriptions of the game found online as described in Section 4.

We use the parameters of the question-answering system to pre-train portions of the deep Q-network for a different game within in the same domain. The portions that are pre-trained are the same parts of the architecture as in Ammanabrolu and Riedl (2019). This game is referred to as the **source task**. The seeding of the knowledge graph is not strictly necessary but given that state-of-the-art DRL agents cannot complete real games, this makes the agent more effective at the source task.

We then transfer the knowledge and skills acquired from playing the source task to another game from the same genre—the **target task**. The parameters of the deep Q-network trained on the source game are used to initialize a new deep Q-network for the target task. All the weights indicated in the architecture of KG-DQN as shown in Fig. 1 are transferred. Unlike Rusu et al. (2016), we do not freeze the parameters of the deep Q-network trained on the source task nor use the two networks to jointly make decisions but instead just use it to initialize the parameters of the target task deep Q-network. This is done to account for the fact that although graph embeddings can be transferred between games, the actual graph extracted from a game is non-transferable due to differences in structure between the games.

6 Experiments

We test our system on two separate sets of games in different domains using the Jericho and TextWorld frameworks (Hausknecht et al., 2019a; Côté et al., 2018). The first set of games is "slice of life" themed and contains games that involve mundane tasks usually set in textual descriptions of normal houses. The second set of games is "horror" themed and contains noticeably more difficult games with a relatively larger vocabulary size and action set, non-standard fantasy names, etc. We choose these domains because of the availability of games in popular online gaming communities, the degree of vocabulary overlap within each theme, and overall structure of games in each theme. Specifically, there must be at least three games in each domain: at least one game to train the question-answering system on, and two more to train the parameters of the source and target task deep Q-networks. A summary of the statistics for the games is given in Table 1. Vocabulary overlap

is calculated by measuring the percentage of overlap between a game's vocabulary and the domain's vocabulary, i.e. the union of the vocabularies for all the games we use within the domain. We observe that in both of these domains, the complexity of the game increases steadily from the game used for the question-answering system to the target and then source task games.

We perform ablation tests within each domain, mainly testing the effects of transfer from seeding, oracle-based question-answering, and source-to-target parameter transfer. Additionally, there are a couple of extra dimensions of ablations that we study, specific to each of the domains and explained below. All experiments are run three times using different random seeds. For all the experiments we report metrics known to be important for transfer learning tasks (Taylor and Stone, 2009; Narasimhan et al., 2017): average reward collected in the first 50 episodes (init. reward), average reward collected for 50 episodes after convergence (final reward), and number of steps taken to finish the game for 50 episodes after convergence (steps). For the metrics tested after convergence, we set $\epsilon = 0.1$ following both Narasimhan et al. (2015) and Ammanabrolu and Riedl (2019). We use similar hyperparameters to those reported in (Ammanabrolu and Riedl, 2019) for training the KG-DQN with action pruning, with the main difference being that we use 100 dimensional word embeddings instead of 50 dimensions for the horror genre.

6.1 Slice of Life Experiments

TextWorld uses a grammar to generate similar games. Following Ammanabrolu and Riedl (2019), we use TextWorld's "home" theme to generate the games for the question-answering system. TextWorld is a framework that uses a grammar to randomly generate game worlds and quests. This framework also gives us information such as instructions on how to finish the quest, and a list of actions that can be performed at each step based on the current world state. We do not let our agent access this additional solution information or admissible actions list. Given the relatively small quest length for TextWorld games—games can be completed in as little as 5 steps—we generate 50 such games and partition them into train and test sets in a 4:1 ratio. The traces are generated on the training set, and the question-answering system is

Table 1: Game statistics

	Slice of life		Horror		
	QA/Source	Target	QA	Source	Target
	TextWorld	*9:05*	*Lurking Horror*	*Afflicted*	*Anchorhead*
Vocab size	788	297	773	761	2256
Branching factor	122	677	-	947	1918
Number of rooms	10	7	25	18	28
Completion steps	5	25	289	20	39
Words per obs.	65.1	45.2	68.1	81.2	114.2
New triples per obs.	6.4	4.1	-	12.6	17.0
% Vocab overlap	19.70	21.45	22.80	14.40	66.34
Max. aug. reward	5	27	-	21	43
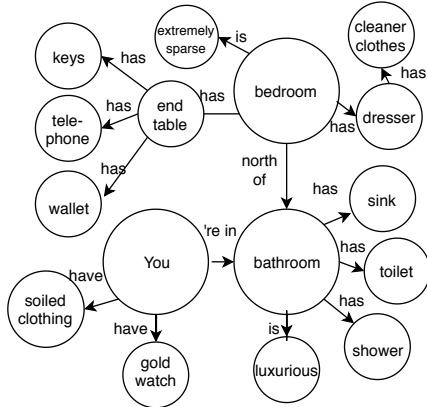
Bedroom
This bedroom is extremely spare, with dirty laundry scattered haphazardly all over the floor. Cleaner clothing can be found in the dresser. A bathroom lies to the south, while a door to the east leads to the living room. On the end table are a telephone, a wallet and some keys.
>*inventory*
You are carrying:
  some soiled clothing (being worn)
  a gold watch (being worn)
>*go south*
Bathroom
This is a far from luxurious but still quite functional bathroom, with a sink, toilet and shower. The bedroom lies to the north.
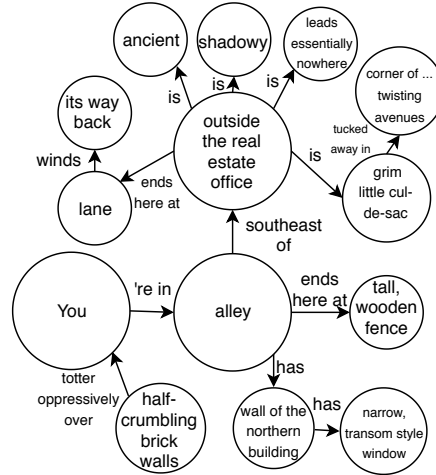
Figure 3: Partial unseeded knowledge graph example given observations and actions in the game *9:05*.

Outside the Real Estate Office
A grim little cul-de-sac, tucked away in a corner of the claustrophobic tangle of narrow, twisting avenues that largely constitute the older portion of Anchorhead. Like most of the streets in this city, it is ancient, shadowy, and leads essentially nowhere. The lane ends here at the real estate agent's office, which lies to the east, and winds its way back toward the center of town to the west. A narrow, garbage-choked alley opens to the southeast.
>*go southeast*
Alley
This narrow aperture between two buildings is nearly blocked with piles of rotting cardboard boxes and overstuffed garbage cans. Ugly, half-crumbling brick walls to either side totter oppressively over you. The alley ends here at a tall, wooden fence. High up on the wall of the northern building there is a narrow, transom-style window.

Figure 4: Partial unseeded knowledge graph example given observations and actions in the game *Anchorhead*.

evaluated on the test set.

We then pick a random game from the test set to train our source task deep Q-network for this domain. For this training, we use the reward function provided by TextWorld: +1 for each action taken that moves the agent closer to finishing the quest; -1 for each action taken that extends the minimum number of steps needed to finish the quest from the current stage; 0 for all other situations.

We choose the game, *9:05*[3] as our target task game due to similarities in structure in addition to the vocabulary overlap. Note that there are multiple possible endings to this game and we pick the simplest one for the purpose of training our agent.

6.2 Horror Experiments

For the horror domain, we choose *Lurking Horror*[4] to train the question-answering system on. The source and target task games are chosen as *Afflicted*[5] and *Anchorhead*[6] respectively. However, due to the size and complexity of these two games some modifications to the games are required for the agent to be able to effectively solve them.

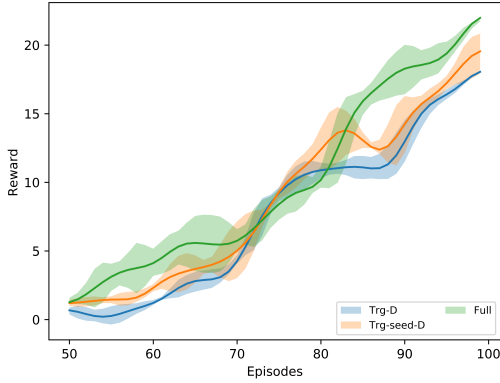[3] https://ifdb.tads.org/viewgame?id=qzftg3j8nh5f34i2

[4] https://ifdb.tads.org/viewgame?id=jhbd0kja1t57uop

[5] https://ifdb.tads.org/viewgame?id=epl4q2933rczoo9x

[6] https://ifdb.tads.org/viewgame?id=op0uw1gn1tjqmjt7

Figure 5: Reward curve for select experiments in the slice of life domain.

Figure 6: Reward curve for select experiments in the horror domain.

We partition each of these games and make them smaller by reducing the final goal of the game to an intermediate checkpoint leading to it. This checkpoints were identified manually using walkthroughs of the game; each game has a natural intermediate goal. For example, *Anchorhead* is segmented into 3 chapters in the form of objectives spread across 3 days, of which we use only the first chapter. The exact details of the games after partitioning is described in Table 1. For Lurking Horror, we report numbers relevant for the oracle walkthrough. We then pre-prune the action space and use only the actions that are relevant for the sections of the game that we have partitioned out. The majority of the environment is still available for the agent to explore but the game ends upon completion of the chosen intermediate checkpoint.

6.3 Reward Augmentation

The combined state-action space for a commercial text-adventure game is quite large and the corresponding reward function is very sparse in comparison. The default, implied reward signal is to receive positive value upon completion of the game, and no reward value elsewhere. This is problematic from an experimentation perspective as text-adventure games are too complex for even state-of-the-art deep reinforcement learning agents to complete. Even using transfer learning methods, a sparse reward signal usually results in ineffective exploration by the agent.

To make experimentation feasible, we augment the reward to give the agent a dense reward signal. Specifically, we use an oracle to generate state-action traces (identical to how as when train-
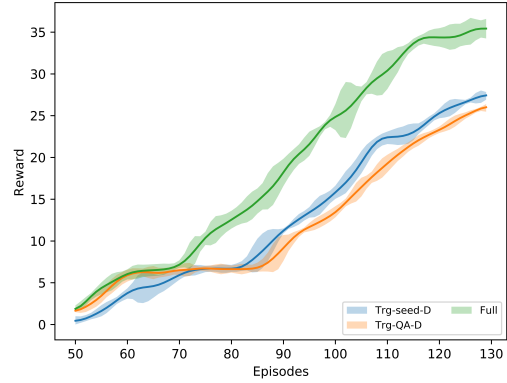
ing the question-answering system). An oracle is an agent that is capable of playing and finishing a game perfectly in the least number of steps possible. The state-action pairs generated using perfect walkthroughs of the game are then used as checkpoints and used to give the agent additional reward. If the agent encounters any of these state-action pairs when training, i.e. performs the right action given a corresponding state, it receives a proportional reward in addition to the standard reward built into the game. This reward is scaled based on the game and is designed to be less than the smallest reward given by the original reward function to prevent it from overpowering the built-in reward. We refer to agents using this technique as having "dense" reward and "sparse" reward otherwise. The agent otherwise receives no information from the oracle about how to win the game.

7 Results/Discussion

The structure of the experiments are such that the for each of the domains, the target task game is more complex that the source task game. The slice of life games are also generally less complex than the horror games; they have a simpler vocabulary and a more linear quest structure. Additionally, given the nature of interactive fiction games, it is nearly impossible—even for human players—to achieve completion in the minimum number of steps (as given by the steps to completion in Table 1); each of these games are puzzle based and require extensive exploration and interaction with various objects in the environment to complete.

Table 2 and Table 3 show results for the slice of life and horror domains, respectively. In both do-

7

Table 2: Results for the slice of life games. "KG-DQN Full" refers to KG-DQN when seeded first, trained on the source, then transferred to the target. All experiment with QA indicate pre-training. S, D indicate sparse and dense reward respectively.

Experiment	Init. Rwd.	Final Rwd.	Steps
Source Game (*TextWorld*)			
KG-DQN no transfer	2.6 ± 0.73	4.7 ± 0.23	110.83 ± 4.92
KG-DQN w/ QA	2.8 ± 0.61	4.9 ± 0.09	88.57 ± 3.45
KG-DQN seeded	3.2 ± 0.57	4.8 ± 0.16	91.43 ± 1.89
Target Game (*9:05*)			
KG-DQN untuned (D)	-	2.5 ± 0.48	1479.0 ± 22.3
KG-DQN no transfer (S)	-	-	1916.0 ± 33.17
KG-DQN no transfer (D)	0.8 ± 0.32	16.5 ± 1.58	1267.2 ± 7.5
KG-DQN w/ QA (S)	-	-	1428.0 ± 11.26
KG-DQN w/ QA (D)	1.3 ± 0.24	17.4 ± 1.84	1127.0 ± 31.22
KG-DQN seeded (D)	1.4 ± 0.35	16.7 ± 2.41	1393.33 ± 26.5
KG-DQN Full (D)	2.7 ± 0.65	19.7 ± 2.0	274.76 ± 21.45

Table 3: Results for horror games. Note that the reward type is dense for all results. "KG-DQN Full" refers to KG-DQN seeded, transferred from source. All experiment with QA indicate pre-training.

Experiment	Init. Rwd.	Final Rwd.	Steps
Source Game (*Afflicted*)			
KG-DQN no transfer	3.0 ± 1.3	14.1 ± 1.73	1934.7 ± 85.67
KG-DQN w/ QA	4.3 ± 1.34	15.1 ± 1.60	1179 ± 32.07
KG-DQN seeded	4.1 ± 1.19	14.6 ± 1.26	1125.3 ± 49.57
Target Game (*Anchorhead*)			
KG-DQN untuned	-	3.8 ± 0.23	-
KG-DQN no transfer	1.0 ± 0.34	6.8 ± 0.42	-
KG-DQN w/ QA	3.6 ± 0.91	24.8 ± 0.6	4874 ± 90.74
KG-DQN seeded	1.7 ± 0.62	26.6 ± 0.42	4937 ± 42.93
KG-DQN full	4.1 ± 0.9	39.9 ± 0.53	4334.3 ± 56.13

mains seeding and QA pre-training improve performance by similar amounts from the baseline on both the source and target task games. A series of t-tests comparing the results of the pre-training and graph seeding with the baseline KG-DQN show that all results are significant with $p < 0.05$. Both the pre-training and graph seeding perform similar functions in enabling the agent to explore more effectively while picking high utility actions.

Even when untuned, i.e. evaluating the agent on the target task after having only trained on the source task, the agent shows better performance than training on the target task from scratch using the sparse reward. As expected, we see a further gain in performance when the dense reward function is used for both of these domains as well. In the horror domain, the agent fails to converge to a state where it is capable of finishing the game without the dense reward function due to the horror games being more complex.

When an agent is trained using on just the target task horror game, *Anchorhead*, it does not converge to completion and only gets as far as achieving a reward of approximately 7 (max. observed reward from the best model is 41). This corre-

sponds to a point in the game where the player is required to use a term in an action that the player has never observed before, "look up Verlac" when in front of a certain file cabinet—"Verlac" being the unknown entity. Without seeding or QA pre-training, the agent is unable to cut down the action space enough to effectively explore and find the solution to progress further. The relative effectiveness of the gains in initial reward due to seeding appears to depend on the game and the corresponding static text document. In all situations except Anchohead, seeding provides comparable gains in initial reward as compared to QA — there is no statistical difference between the two when performing similar t-tests.

When the full system is used—i.e. we seed the knowledge graph, pre-train QA, then train the source task game, then the target task game using the augmented reward function—we see a significant gain in performance, up to an 80% gain in terms of completion steps in some cases. The bottleneck at reward 7 is still difficult to pass, however, as seen in Fig. 6, in which we can see that the agent spends a relatively long time around this reward level unless the full transfer technique is

used. We further see in Figures 5, 6 that transferring knowledge results in the agent learning this higher quality policy much faster. In fact, we note that training a full system is more efficient than just training the agent on a single task, i.e. training a QA system then a source task game for 50 episodes then transferring and training a seeded target task game for 50 episodes is more effective than just training the target task game by itself for even 150+ episodes.

8 Conclusions

We have demonstrated that using knowledge graphs as a state representation enables efficient transfer between deep reinforcement learning agents designed to play text-adventure games, reducing training times and increasing the quality of the learned control policy. Our results show that we are able to extract a graph from a general static text resource and use that to give the agent knowledge regarding domain specific vocabulary, object affordances, etc. Additionally, we demonstrate that we can effectively transfer knowledge using deep Q-network parameter weights, either by pre-training portions of the network using a question-answering system or by transferring parameters from a source to a target game. Our agent trains faster overall, including the number of episodes required to pre-train and train on a source task, and performs up to 80% better on convergence than an agent not utilizing these techniques.

We conclude that knowledge graphs enable transfer in deep reinforcement learning agents by providing the agent with a more explicit–and interpretable–mapping between the state and action spaces of different games. This mapping helps overcome the challenges twin challenges of partial observability and combinatorially large action spaces inherent in all text-adventure games by allowing the agent to better explore the state-action space.

9 Acknowledgements

References

Prithviraj Ammanabrolu and Mark O. Riedl. 2019. Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*.

Gabor Angeli, Johnson Premkumar, Melvin Jose, 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)*.

Antoine Bordes, Nicolas Usunier, Ronan Collobert, and Jason Weston. 2010. Towards understanding situated natural language. In *Proceedings of the 2010 International Conference on Artificial Intelligence and Statistics*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. TextWorld : A Learning Environment for Text-based Games. In *Proceedings of the ICML/IJCAI 2018 Workshop on Computer Games*, page 29.

Nancy Fulda, Daniel Ricks, Ben Murdoch, and David Wingate. 2017. What can you do with a rock? affordance extraction via word embeddings. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1039–1045.

Milica Gasic, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve J. Young. 2013. Pomdp-based dialogue manager adaptation to extended domains. In *SIGDIAL Conference*.

Matan Haroush, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. 2018. Learning How Not to Act in Text-Based Games. In *Workshop Track at ICLR 2018*, pages 1–4.

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

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

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Li-hong Li, Li Deng, and Mari Ostendorf. 2016. Deep Reinforcement Learning with a Natural Language Action Space. In *Association for Computational Linguistics (ACL)*.

Girish Joshi and Girish Chowdhary. 2018. Cross-domain transfer in reinforcement learning using target apprentice. In *Proceedings of the International Conference on Robotics and Automation*, pages 7525–7532.

George Konidaris and Andrew G. Barto. 2007. Building portable options: Skill transfer in reinforcement learning. In *IJCAI*.

George Konidaris, Ilya Scheidwasser, and Andrew G. Barto. 2012. Transfer in reinforcement learning via shared features. *The Journal of Machine Learning Research*, 13:1333–1371.

Long-Ji Lin. 1993. *Reinforcement learning for robots using neural networks*. Ph.D. thesis, Carnegie Mellon University.

Yaxin Liu and Peter Stone. 2006. Value-function-based transfer for reinforcement learning using structure mapping. In *AAAI*.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2017. Deep transfer in reinforcement learning by language grounding. *Journal of Artificial Intelligence Research*, 63.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language Understanding for Text-based Games Using Deep Reinforcement Learning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Trung Thanh Nguyen, Tomi Silander, and Tze-Yun Leong. 2012. Transferring expectations in model-based reinforcement learning. In *NIPS*.

Emilio Parisotto, Jimmy Ba, and Ruslan R. Salakhutdinov. 2016. Actor-mimic: Deep multitask and transfer reinforcement learning. *CoRR*, abs/1511.06342.

Janarthanan Rajendran, Aravind S. Lakshminarayanan, Mitesh M. Khapra, P. Prasanna, and Balaraman Ravindran. 2017. Attend, adapt and transfer: Attentive deep architecture for adaptive transfer from multiple sources in the same domain. In *ICLR*.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *CoRR*, abs/1606.04671.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT Press.

Ruo Yu Tao, Marc-Alexandre Côté, Xingdi Yuan, and Layla El Asri. 2018. Towards solving text-based games by producing adaptive action spaces. In *Proceedings of the 2018 NeurIPS Workshop on Wordplay: Reinforcement and Language Learning in Text-based Games*.

Matthew E. Taylor, Nicholas K. Jong, and Peter Stone. 2008. Transferring instances for model-based reinforcement learning. In *ECML/PKDD*.

Matthew E. Taylor and Peter Stone. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685.

Zhuoran Wang, Tsung-Hsien Wen, Pei hao Su, and Yannis Stylianou. 2015. Learning domain-independent dialogue policies via ontology parameterisation. In *SIGDIAL Conference*.

H. Yin and S. J. Pan. 2017. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 1640–1646. AAAI Press.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter Battaglia. 2019. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations*.