Grounding Natural Language with Autonomous Interaction

by

Karthik Rajagopal Narasimhan

B.Tech., Indian Institute of Technology, Madras (2012)
S.M., Massachusetts Institute of Technology (2014)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2017

Author . . . .
Department of Electrical Engineering and Computer Science
August 24, 2017

Certified by . . . .
Regina Barzilay
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair of the Committee on Graduate Students

Grounding Natural Language with Autonomous Interaction

by

Karthik Rajagopal Narasimhan

Submitted to the Department of Electrical Engineering and Computer Science
on August 24, 2017, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

The resurgence of deep neural networks has resulted in impressive advances in natural language processing (NLP). This success, however, is contingent on access to large amounts of structured supervision, often manually constructed and unavailable for many applications and domains. In this thesis, I present novel computational models that integrate reinforcement learning with language understanding to induce grounded representations of semantics. Using unstructured feedback, these techniques not only enable task-optimized representations which reduce dependence on high quality annotations, but also exploit language in adapting control policies across different environments.

First, I describe an approach for learning to play text-based games, where all interaction is through natural language and the only source of feedback is in-game rewards. Employing a deep reinforcement learning framework to jointly learn state representations and action policies, our model outperforms several baselines on different domains, demonstrating the importance of learning expressive representations.

Second, I exhibit a framework for utilizing textual descriptions to tackle the challenging problem of cross-domain policy transfer for reinforcement learning (RL). We employ a model-based RL approach consisting of a differentiable planning module, a model-free component and a factorized state representation to effectively make use of text. Our model outperforms prior work on both transfer and multi-task scenarios in a variety of different environments.

Finally, I demonstrate how reinforcement learning can enhance traditional NLP systems in low resource scenarios. In particular, I describe an autonomous agent that can learn to acquire and integrate external information to enhance information extraction. Our experiments on two databases – shooting incidents and food adulteration cases – demonstrate that our system significantly improves over traditional extractors and a competitive meta-classifier baseline.

Thesis Supervisor: Regina Barzilay
Title: Professor of Electrical Engineering and Computer Science

Acknowledgements

This thesis is the culmination of an exciting five-year journey and I am grateful for the support, guidance and love of several mentors, colleagues, friends and family members. The most amazing thing about MIT is the people here – brilliant, visionary, and dedicated to pushing the boundaries of science and technology. A perfect example is my advisor, Regina Barzilay. Her enthusiasm, vision and mentorship have played a huge role in my PhD journey and I am forever indebted to her for helping me evolve into a competent researcher.

I would like to thank my wonderful thesis committee of Tommi Jaakkola and Luke Zettlemoyer. In addition to key technical insights, Tommi has always patiently provided sound advice, both research and career related. Luke has been a great mentor, and much of his research was a big source of inspiration during my initial exploration into computational semantics.

I have also had some great mentors over the last few years. In particular, SRK Branavan and Tao Lei provided valuable advice during my initial years into the program. Dipanjan Das provided valuable research and career advice during my time at Google. I also wish to thank my amazing collaborators – Tejas Kulkarni, Ardavan Saeedi, Adam Yala, Jiaming Luo, Michael Janner, Damianos Karakos, Richard Schwartz, Stavros Tsakalidis, Jonathan Huggins, Kayhan Batmanghelich, Nicholas Locascio, Eduardo DeLeon, Nate Kushman, Josh Tenenbaum, Sam Gershman and my advisor Regina Barzilay.

My labmates, both current and former, have been a big part of my life at MIT, ranging from intellectual research discussions to fun social events – thanks to Tahira Naseem, Yoong Keok Lee, Yevgeni Berzak, Yonatan Belinkov, Zach Hynes, Yuan Zhang, Darsh Shah, Wengong Jin, Tianxiao Shen, Benson Chen and Abdi Dirie. I would also like to acknowledge our group's administrative assistant, Marcia Davidson, for patiently helping me out with any issue, big or small. Special thanks to the CSAIL and EECS communities at MIT for fostering the perfect research atmosphere, balanced with ample opportunity for social interaction.

I have been extremely lucky to have wonderful family and friends who have supported me through this journey. My parents and brother – Uma, Narasimhan and Bharath – have always encouraged me to aim higher and not rest on past achievements. My uncle Narayan and his family helped me adjust to a new country and made sure I always had someone to visit during the holidays. Several close friends – Julian, Diego, Karthick, Mikhil, Ankur, Rushabh, Andreaa, Vikas, Vaibhav, Sanket, Suvinay, Anirudh, Rahul and Yewen – were big sources of support and made sure I had enough play to balance out the workload. Finally, I am grateful to Ramya, whose love and care helped me handle the ups and downs of PhD life. Meeting her was one of the best things that happened to me at MIT.

Bibliographic Note

Portions of this thesis are based on prior peer-reviewed publications. The autonomous player for text-based games, presented in Chapter 2, was originally published in Narasimhan et al. [78]. The system for improving information extraction with reinforcement learning (Chapter 4) was published in Narasimhan et al. [79]. Finally, a version of the technique for transfer in reinforcement learning by grounding language [77] is currently under peer-review.

The code and data for the techniques presented in this thesis are available at `https://github.com/karthikncode`

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A long-standing goal of artificial intelligence has been to enable computers to understand human languages. Progress in this field will increase the ability of autonomous systems to access, manage and exploit the vast amount of human knowledge collected in textual form. An important development in this direction has been the resurgence of deep learning techniques in natural language processing (NLP), bringing about impressive advances in applications like machine translation [67, 7, 65], speech recognition [49, 28] and automated question answering [51, 84, 121]. The success of these neural network approaches, however, is contingent upon access to large amounts of training data, often manually annotated. Large-scale supervision is often quite expensive to obtain, and unobtainable for many tasks and domains. A key research direction, therefore, is the development of NLP techniques that can operate without requiring substantial supervision. In this thesis, I present systems that directly combine semantic interpretation with autonomous interaction in environments, alleviating the need for prohibitive amounts of structured annotation.

Traditional approaches to semantics in computational linguistics have sought to encapsulate the meaning of text into various formal structures. Such formalisms include semantic role labels (SRL) [83], semantic parse trees [8, 62], and first-order expressions [89, 11]. Although recent research has demonstrated noticeable improvements on several of these benchmarks, employing these formalisms effectively in tar-

Face the octagon carpet. Move until you see red brick floor to your right. Turn and walk down the red brick until you get to an alley with grey floor. You should be two alleys away from a lamp and then an easel beyond that. Go forward one segment to the intersection with the pink-flowered carpet.

The brown scorpions will slowly move towards the player. The red bat also chases the player. The pickaxe can be used to dig the earth. Diamonds can be picked up by the player.

Figure 1-1: Examples of text that can influence behavior of an autonomous system in interactive environments. The top box contains 'instructions' that can be directly mapped to actions, while the bottom one consists of 'descriptions' that correspond to dynamics of the environment.

geted applications remains a challenge. Further, there is no clear consensus within the NLP community on the relative merits of these different designs.

In this thesis, I explore an alternative approach to language semantics: grounding textual meaning directly on to control tasks in interactive domains. In these environments, there exists a flow of signals in two directions – an autonomous system takes actions that change the world's state, while receiving feedback based on certain criteria. An agent is considered to have a 'grounded understanding' of language if it can map words and sentences to the objects, dynamics and/or actions in its surroundings. Consider the examples provided in Figure 1-1 for instance. The first piece of text is instructive, providing turn by turn directions on the course of action. Learning a mapping from these words to actions can help the agent successfully navigate the environment. The second piece of text, on the other hand, describes activities in a domain, and is hence more subtle. Here, the semantics can be mapped to the dynamics of the environment, an understanding of which helps the agent plan its actions. In both cases, successful completion of the end task indicates effective interpretation of language.

This form of semantic interpretation provides two benefits. First, the need for manual annotation is eliminated. There is no specific intermediate formalism required, since the evaluation of a system is solely based on its ability to successfully navigate the environment or complete the task. Second, this framework allows for inducing semantic representations that are optimized for the eventual control task. This can be achieved by developing models that permit joint learning of both representations and control policies.

I investigate two directions of integrating language understanding with autonomous interaction. The first scenario involves learning representations for text using rewards from environments. Text comprehension is inherently contextual, and varies for each specific situation. In this setup, we utilize the backdrop provided by the observations in the environment to induce textual representations optimized for the task. The second direction explores the use of knowledge encoded in text to assist policy learning for control applications. The symbolic information carried by words and sentences help connect different aspects of the control domain, enabling agents to learn faster and more accurately. Together, these directions complement each other and form a holistic view of semantic interpretation of language.

We tackle a few concrete challenges in the environments considered:

- **Interpreting compositional language**: While some understanding can be performed at the word level, a complete semantic interpretation often requires parsing higher-order structures in text. For instance, even though they contain the same set of words, the difference between the instructions 'Move to the right, not the left' and 'Move to the left, not the right' is quite significant in terms of their implied actions sequences. Hence, we require an effective mechanism to handle compositional aspects of language such as conjunction or negation.

- **Performing the correct level of semantic mapping**: As demonstrated earlier in Figure 1-1, language can be grounded to various aspects of control, such as actions or environment dynamics. Effectively leveraging the different types of mapping requires an appropriate model representation. Further, in each aspect, we are required to handle multiple levels of abstraction in semantic concepts. Some are low-level (e.g. *If you go east, you will be back on solid* *ground*), while some require world knowledge to infer consequences (e.g. *The bridge sways in the wind.*).

- **Handling Imperfect Text**: There is no guarantee that the information provided in the text is complete. Further, some parts of the text might be irrelevant to the current situation. We require methods that don't overly rely on provided descriptions, but instead integrate useful knowledge derived from text with experience obtained through direct interaction with the environment.

- **Learning semantics from unstructured feedback**: In contrast to supervised learning, the feedback obtained for control policies is in the form of scalar values upon completion (or failure) of the task. This necessitates methods that can leverage such unstructured feedback to learn useful grounding for text.

With these challenges in mind, we model our control tasks as Markov Decision Processes (MDP). We then utilize the framework of Reinforcement Learning (RL) [105] to induce control policies. As opposed to supervised learning, RL does not require input-output pairs for learning, and tackles the problem of delayed feedback using approximate dynamic programming [12]. Further, the stochastic nature of these environments is implicitly accounted for using a value function, which optimizes for expected future reward over all possible outcomes. Moreover, state space approximation techniques in deep reinforcement learning provide us with the ability to generalize well across large state spaces, with far less samples than the entire set of states.

I explore these techniques in the context of three different scenarios:

1. **Language understanding for text-based games**: Understanding natural language is inherently contextual, requiring the perception and comprehension of one's environment for unambiguous interpretation. In spite of recent advancements in several NLP applications, there has been limited work on integrated frameworks for language comprehension grounded in interactive environments. We consider the task of learning to play text-based games [27, 4] without any prior knowledge of language. The agent is provided with a description of its

current state and can ‘act’ by writing text commands to the game environment. The challenges lie in interpreting compositional text, identifying parts relevant for choosing actions, and learning from sparse feedback. We assume no other form of supervision other than the rewards inherent to the game environment.

2. **Policy transfer via language grounding**: Natural language helps encode world knowledge compactly, enabling people to adapt to different situations in their lives. Using this insight, we develop autonomous agents that utilize language to transfer control policies across different domains. Specifically, we consider environments augmented with textual descriptions of their dynamics, detailing movements and interactions of various entities without providing any information about task objectives. In this case, understanding text is an auxiliary requirement; an autonomous agent can learn an optimal policy without doing so. Comprehending the provided text, however, will help the system learn more effectively (especially in large stochastic environments with sparse rewards) and generalize faster to new domains. The main challenges lie in integrating both textual information and experience gained through interaction, handling incomplete textual information, and learning policies across multiple tasks.

3. **Improved information extraction with reinforcement learning**: In the final scenario, we consider the problem of information extraction [41], where the goal is to process large pieces of unstructured text into structured information. Traditional approaches perform poorly when the amount of data for training is limited, mainly because not all forms of expression will be seen during model learning. We develop an autonomous system that learns to gather and aggregate extra information from articles on the web to improve extraction performance. This setting is different from the previous two applications since the entire world wide web is our virtual environment, presenting a huge state space to explore. Other challenges here lie in deciding which external articles are relevant to the event considered and reconciling inconsistent extractions from different

sources. Our novel reinforcement learning formulation provides a way to boost any extraction system by exploiting the redundancy on the web.

I now provide a summary of these three applications and briefly describe how we tackle each one.

1.1 Language understanding for text-based games

We first tackle the problem of developing an autonomous agent to play text-based games. These games, also known as multi-user dungeons (MUDs), were very popular before the advent of graphical games, and still have a large following worldwide [1], All interactions in these games are through text – the player is provided with a description of his current state in the virtual world, and is required to enter a textual command to take an action. Figure 1-2 provides an example interaction.

There are three main challenges in developing an autonomous player specific to this domain:

1. **No symbolic representation**: The most pressing challenge for an autonomous player is to understand the text provided to make sense of the current situation. Without correctly interpreting the text, the agent cannot successfully navigate this domain.

2. **Varying state descriptions**: The text provided for a room varies considerably, with game developers employing multiple ways of providing the same information. Thus, the agent cannot simply memorize a mapping from text to states.

3. **Incomplete information**: The state descriptions often do not contain every piece of information about the current state, in contrast to fully observable graphical games like Pacman. This adds another layer of complication to inducing a successful control policy.

Our goal is to simultaneously learn both a control policy and an optimal representation for the input text. For supervision, we make use of in-game rewards, coinciding with happenings in the game. The reward can be either positive, such as when the player obtains gold or defeats an enemy, or negative, like when the player dies. Even though this type of feedback is quite unstructured, it provides us with the opportunity of learning representations for text that are directly optimized for the end task.

[1] `https://www.mudstats.com`

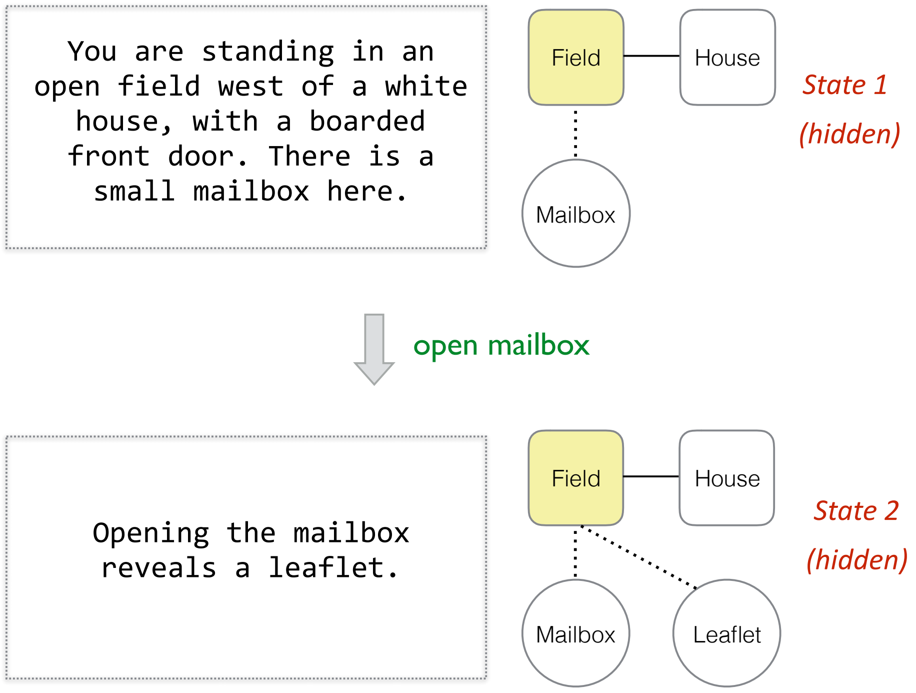

You are standing in an open field west of a white house, with a boarded front door. There is a small mailbox here.
Field
House
Mailbox
State 1
(hidden)
open mailbox
Opening the mailbox reveals a leaflet.
Field
House
Mailbox
Leaflet
State 2
(hidden)

Figure 1-2: Example of a text-based adventure game (*Zork*). The text on the left is what a player gets to observe, while the game engine keeps track of the hidden state. When the player inputs a command, such as OPEN MAILBOX, the game transitions to a different state, and presents the player with the next component of the narrative.

We formalize the task using the framework of reinforcement learning, where the input text is treated as a state, and the commands are the actions that an agent can take. We then learn an action policy in order to choose the best possible commands under different circumstances.

Through experiments on multiple games we demonstrate that our model learns to successfully interpret text and navigate through text-based adventures, outperforming several baselines. Moreover, we show that the acquired representation can be reused across games, speeding up learning.

1.2 Policy transfer via language grounding

In the second application, we utilize natural language to enable transfer for reinforcement learning. This work is motivated by a long-standing goal of RL to learn universal policies that apply across related tasks, in contrast to current methods that have limited transfer capabilities. Although deep reinforcement learning has been successfully applied to many tasks [73, 100, 64], its success is contingent upon a significant amount of interactions with the environment. This inefficiency has spurred recent work in transfer methods [85, 91, 31, 112, 92, 93] and generalized value functions [97]. However, a key challenge in performing successful transfer lies in obtaining an effective mapping between domains (e.g. state or action spaces), which requires a non-trivial amount of exploration in a new environment.

I propose a novel approach to transfer in reinforcement learning: utilize text descriptions to facilitate policy generalization across domains. Natural language plays an important role in human exploration of the world, enabling compact transfer of knowledge and helping people generalize from their own and each other's experiences. For instance, if a person were to find herself in unseen territory (say, the moon), she could make use of information about the environment such as "gravitation of the moon is 1/6th that of the earth" to pre-adapt known policies like jumping and running appropriately. Along the same lines, language can act as a bridge to transfer generic policies across domains, helping constrain the space of dynamics an autonomous agent has to reason about in a new environment.

Consider the two game environments in Figure 1-3 along with sample descriptions. The two games – *Boulderchase* and *Freeway* – differ in their layouts and entity types. However, the high-level behavior of most entities in both games is similar. For instance, the *diamond* in Boulderchase is a motionless entity, similar to the *tree* in Freeway. Though this similarity is clearly reflected in the text descriptions in Figure 1-3, it may require multiple interactions with the environment for the agent to discover. Therefore, exploiting these textual clues could help an autonomous agent learn an optimal policy faster.

We demonstrate that by learning to ground the meaning of the text to dynamics of the environment such as transitions and rewards, an agent can effectively bootstrap policy learning on a new domain. To achieve this, there are two key requirements of a model - (1) to effectively incorporate (noisy) text into the action-value prediction, and (2) to learn the transition and reward functions of the world through only interaction. We satisfy both by combining a factorized representation generator with a model-aware value iteration network (VIN) [108]. Both modules are parameterized as deep neural networks and fit together to form a single end-to-end differentiable model. We demonstrate strong transfer and multi-task results, outperforming several baselines and a state-of-the-art transfer approach on various domains.

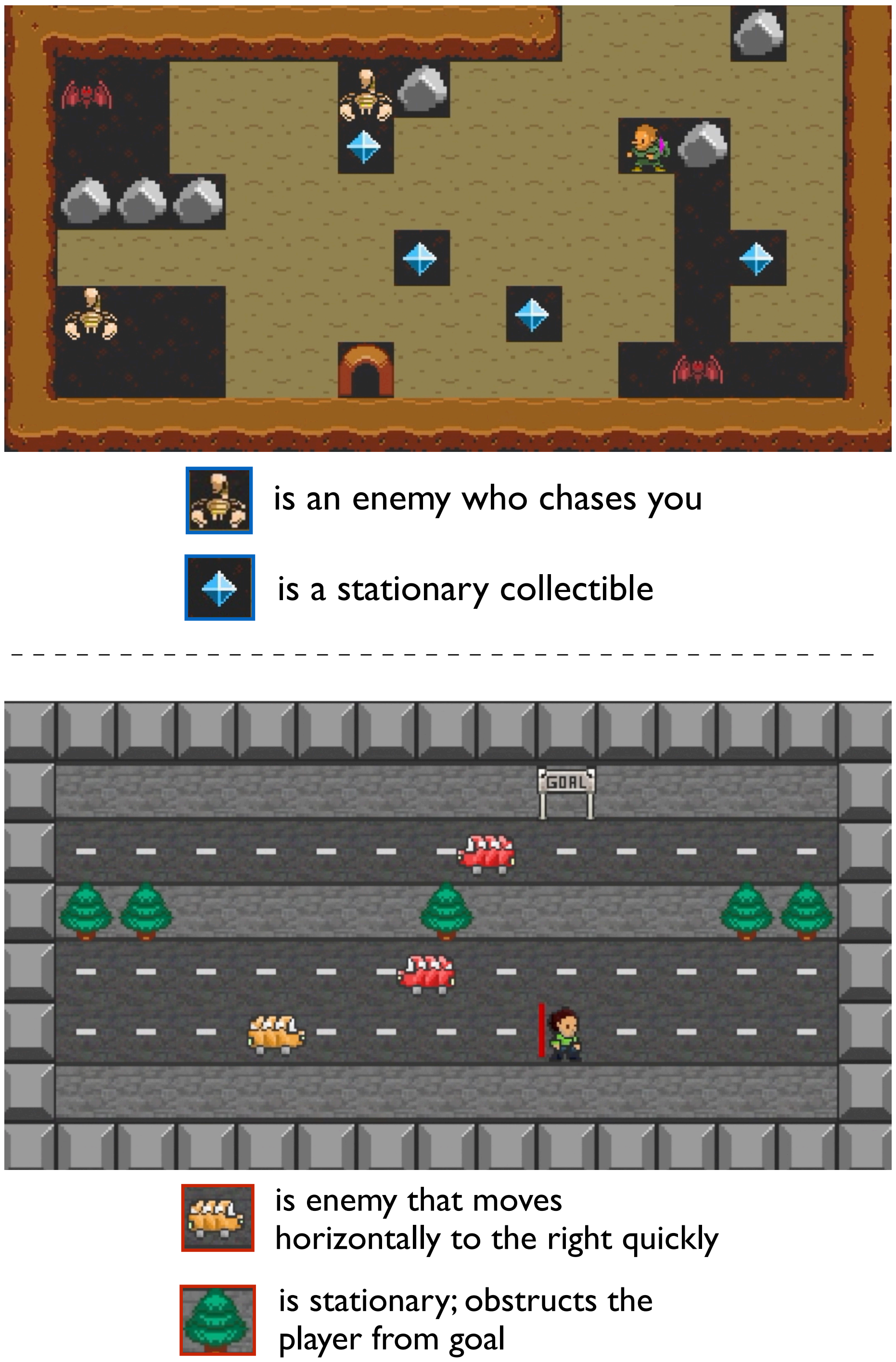

is an enemy who chases you
is a stationary collectible
GOAL
is enemy that moves horizontally to the right quickly
is stationary; obstructs the player from goal

Figure 1-3: Two different game environments: Boulderchase **(top)** and Freeway **(bottom)**, along with associated descriptions in text. Some entities across these games have similar behaviors. For instance, both the *diamond* in Boulderchase and the *tree* in Freeway are motionless objects. This fact might take the agent several episodes of interaction to discover, but is clearly discernible from the text.

1.3 Improved information extraction with reinforcement learning

In the third application, we tackle the problem of information extraction (IE), a traditional NLP task. Most IE systems require substantial amounts of supervision, with their performance rapidly diminishing with decreases in training data size. This is mainly because the same fact can be expressed using a variety of different linguistic forms, all of which cannot be observed during training. Hence, when a new pattern is shown during testing, the system has difficulty in reliably extracting the required information. However, this same information might be expressed in a more 'understandable' fashion in a different piece of text.

To this end, we propose a system that learns to query and aggregate information from an external source, like the world wide web, in an autonomous fashion (Figure 1-4). Given a document and a base extractor to make use of, the model learns to first search for other articles on the same event from web. Subsequently, the system applies the base extractor on the retrieved articles, and intelligently aggregates information extracted from the different sources. This process is repeated until sufficient evidence is collected. This is very similar to how humans assimilate information. When a person is given a document she does not understand or does not get the required details from, she would resort to finding other articles on the same topic to gather more information.

We encounter two main challenges – (1) event co-reference, which requires us to identify articles on the same event, and (2) entity resolution, where we need a mechanism to reconcile the varying predictions from different articles. To address these challenges, we formulate the information extraction process in a reinforcement learning framework, using prediction accuracies as a source of rewards. Both article selection and value reconciliation are actions available to our system to select at each step. We demonstrate the effectiveness of our approach on two different domains, with significant performance gains over base extractors and aggregation baselines.

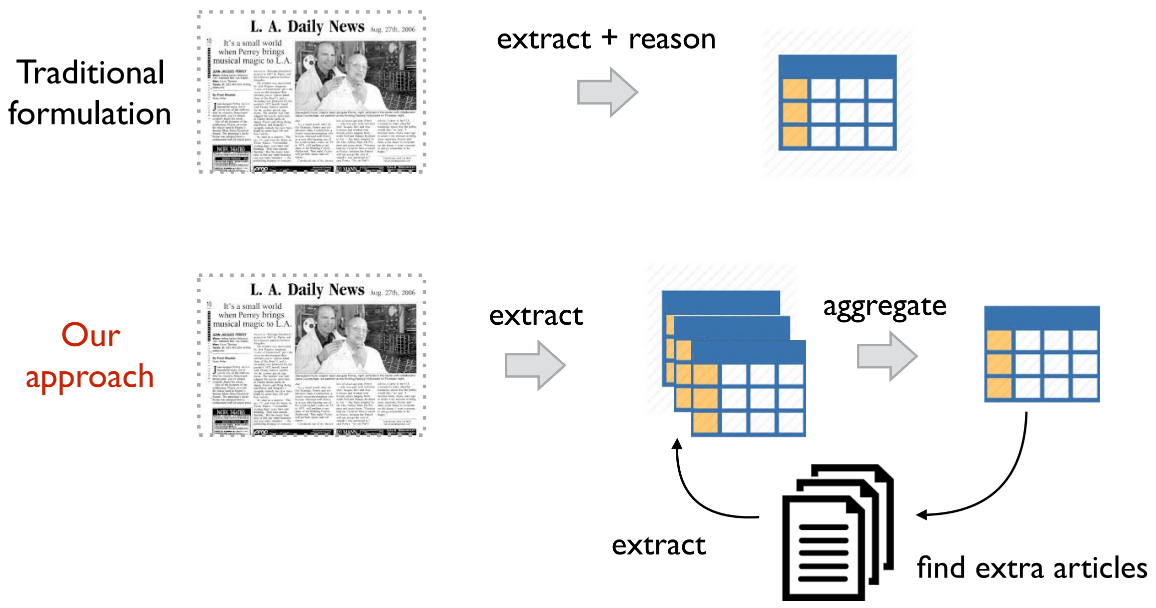


Figure 1-4: Overview of our approach to information extraction. While traditional models perform a single extraction (plus any additional reasoning) from the original article, our method searches for other related articles on the web and aggregates information spread across these documents. We build an RL model that learns to perform both querying and value reconciliation, without need for any extra annotations. As our experiments demonstrate, our system obtains significant performance gains over traditional systems and simple aggregation baselines.

1.4 Contributions

The primary contributions of this thesis are threefold:

- **Learning representations for language to navigate text-based games**: Given access to an interactive environment such as a text adventure game, we demonstrate how to effectively learn task-optimized representations for natural language without requiring any manual annotations. With unstructured feedback from the environment, we also simultaneously learn a control policy that can utilize this representation to navigate the domain.

- **Leveraging language to enable policy transfer**: We explore the utility of language as a compact representation of knowledge that enables cross-domain transfer for reinforcement learning. As we demonstrate empirically, access to even a small amount of textual descriptions results in substantially improved learning for new domains.

- **Autonomous exploration and aggregation of evidence to improve information extraction**: We propose a novel framework for improving information extraction by autonomously querying, retrieving and aggregating external evidence from the world wide web. Leveraging the redundant information present across articles on the same event, we demonstrate more accurate and reliable extraction, especially in low-resource scenarios. This technique can also be adapted to other related NLP tasks like question answering or automated dialogue agents.

Together, these pieces of work represent a step towards a tighter coupling of natural language interpretation and action policies for interactive environments. We show how each realm can benefit from the other – the context provided by the environment can aid robust text understanding, while the knowledge encoded in text can help improve performance on control applications. These models open up possibilities for systems that can ground language onto aspects of their environment, resulting in a more direct utilization of their understanding.

1.5 Outline

The rest of this thesis is organized as follows:

- **Chapter 2** presents an approach to learn automated agents for text-based adventure games using only in-game rewards. Using deep reinforcement learning, the proposed model acquires representations that allow it to successfully navigate these narratives.

- **Chapter 3** explores a novel transfer method for deep reinforcement learning, leveraging knowledge encoded in textual descriptions of environments. Utilizing a model-aware approach, the method results in faster and more efficient learning on new, unseen domains.

- **Chapter 4** details our system for autonomously obtaining and incorporating external evidence to improve information extraction. Without need for additional supervision, the system is trained using reinforcement learning and provides substantial improvements in extraction accuracy over several baselines.

- **Chapter 5** concludes the thesis, summarizing the key points and providing a few directions for future work.

Chapter 2

Language Understanding for Text-based Games

In this chapter, we consider the task of learning control policies for text-based games. In these games, all interactions in a virtual world are through text and the underlying state is not observed. The resulting language barrier makes such environments challenging for automatic game players. We employ a deep reinforcement learning framework to jointly learn state representations and action policies using game rewards as feedback. This enables us to map text descriptions into vector representations that capture the semantics of the game states. This expressive representation forms the basis for effective policy learning, as we demonstrate through our experiments.

2.1 Introduction

Text-based adventure games, predecessors to modern graphical ones, still enjoy a large following worldwide.[1] These games often involve complex worlds with rich interactions and elaborate textual descriptions of the underlying states (see Figure 2-1). Players read descriptions of the current world state and respond with natural language commands to take actions. Since the underlying state is not directly observable, a

[1] `http://mudstats.com/`

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 …
Under your feet a plank comes loose, tumbling down. For a moment you dangle over the abyss …
Below you a particularly large wave crashes into the rocks.

Command: GO EAST

State 2: Ruined gatehouse

The old gatehouse is near collapse. Part of its northern wall has already fallen down, together with parts of the fortifications in that direction. Heavy stone pillars hold up sections of ceiling, but elsewhere the flagstones are exposed to open sky. Part of a heavy portcullis, formerly blocking off the inner castle from attack, is sprawled over the ground together with most of its frame.
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 2-1: Sample gameplay from a text-based fantasy adventure. The player has the quest of finding a secret tomb, and is currently located on an unstable *old bridge* (State 1). If the wind blows hard enough, the player will tumble down into the valley. She chooses an action by typing the command GO EAST, which brings her to a *ruined gatehouse* (State 2). Note how each state description is quite long and requires interpreting several aspects (and ignoring others) to respond appropriately.

player has to understand the text in order to act, making it challenging for existing AI programs to play these games [29].

In designing an autonomous game player, we have considerable latitude when selecting an adequate state representation to employ. The simplest method would be to use a bag-of-words representation derived from the text description. However, this scheme disregards the ordering of words and the finer nuances of meaning that evolve from composing words into sentences and paragraphs. For instance, in *State 2* in Figure 2-1, the agent has to understand that going *east* will lead it to the castle whereas moving *south* will take it to the standing archway. An alternative approach is to convert text descriptions to pre-specified representations using annotated training data, which is commonly used in language grounding tasks [69, 61].

In contrast, our goal is to learn useful representations for text in conjunction with control policies. We adopt a reinforcement learning framework and formulate game sequences as Markov Decision Processes. In this setup, an agent aims to maximize rewards that it obtains from the game engine upon the occurrence of certain events (such as finding treasure or defeating an enemy). The agent learns a policy in the form of an action-value function $Q(s, a)$ which denotes the long-term merit of an action a in state s.

The action-value function is parametrized using a deep recurrent neural network, trained using the game feedback. The network contains two modules. The first one converts textual descriptions into vector representations that act as proxies for states. This component is implemented using Long Short-Term Memory (LSTM) networks [50]. The second module of the network scores the actions given the vector representation computed by the first. This component is realized as a deep Q-network (DQN) [73]. Together, both modules form a single end-to-end differentiable network.

We evaluate our model using two Multi-User Dungeon (MUD) games [27, 4]. The first game is designed to provide a controlled setup for the task, while the second is a publicly available one containing human generated text descriptions with significant language variability. We compare our algorithm against several baselines, including a random player and models that represent the state as a bag of words or bigrams. We demonstrate that our model, LSTM-DQN, significantly outperforms these baselines in terms of number of completed quests and accumulated rewards. For instance, on a fantasy MUD game, our model learns to complete 96% of the quests, while the bag-of-words model and a random baseline solve only 82% and 5% of the quests, respectively. Moreover, we show that the acquired representation can be reused across game instances, speeding up learning and leading to faster convergence of policies.

The remainder of this chapter is organized as follows. We first present a summary of related work in Section 2.2. In Section 2.3, we provide some background on the framework and methods we use in our system. Section 2.4 details our model architecture, while Section 2.5 describes our learning algorithm. We provide experimental details in Section 2.6, followed by empirical results and analysis in Section 2.7. Section 2.8 concludes the chapter and provides directions for future research.

2.2 Related Work

2.2.1 Leveraging text for control applications

Learning control policies using text is gaining increasing interest in the NLP community. Example applications include interpreting help documentation for software [16], navigating with directions [114, 56, 6, 69, 5] and playing computer games [33, 14]. The bulk of these approaches utilize linguistic analysis techniques to structure the information contained in text, which can then be utilized by task-specific control policies. This requires choosing a specific intermediate semantic representation for text, which might vary depending on the task domain. Our approach, on the other hand, does not use an explicit intermediate structure, instead learning vector representations optimized for the control task.

2.2.2 Linguistic analysis for games

Games provide a rich domain for grounded language analysis. Prior work has assumed perfect knowledge of the underlying state of the game to learn policies. Gorniak and Roy [40] developed a game character that can be controlled by spoken instructions adaptable to the game situation. The grounding of commands to actions is learned from a transcript manually annotated with actions and state attributes. Eisenstein et al. [33] learn game rules by analyzing a collection of game-related documents and precompiled traces of the game. In contrast to the above work, our model combines text interpretation and strategy learning in a single framework. As a result, textual analysis is guided by the received control feedback, and the learned strategy directly builds on the text interpretation.

Our work closely relates to an automatic player that utilizes text manuals to learn strategies for the game of Civilization [14]. Similar to our approach, text analysis and control strategies are learned jointly using feedback provided by the game simulation. In their setup, states are fully observable, and the model learns a strategy by combining state/action features and features extracted from text. However, in our application, the state representation is not provided, but has to be inferred from a textual description. Therefore, it is not sufficient to extract features from text to supplement a simulation-based player.

2.2.3 Deep reinforcement learning for game playing

Another related line of work consists of automatic video game players that infer state representations directly from raw pixels [59, 73]. For instance, Mnih et al. [73] learn control strategies using convolutional neural networks, trained with a variant of Q-learning [117]. While both approaches use deep reinforcement learning for training, our work has important differences. In order to handle the sequential nature of text, we use Long Short-Term Memory networks to automatically learn useful representations for arbitrary text descriptions. Additionally, we show that decomposing the network into a representation layer and an action selector is useful for transferring the learned representations to new game scenarios.

Recent work [46] has built upon the work presented in this chapter to handle longer text responses from the player to the game engine. The proposed model makes a choice between predefined responses, by learning a representation for each response and scoring it conditioned on the current state description.

2.3 Background

2.3.1 Game Representation

We represent a game by the tuple $\langle H, A, T, R, \Psi \rangle$, where H is the set of all possible game states, A is the set of all commands, $T(h' \mid h, a)$ is the stochastic transition function between states and $R(h, a)$ is the reward function. Each command $a \in A$, is a two-word phrase $(\bar{a}, \bar{o})$ consisting of an action word $\bar{a}$ and an object $\bar{o}$. The game state H is *hidden* from the player, who only receives a varying textual description, produced by a stochastic function $\Psi : H \rightarrow S$. Specifically, the underlying state h in the game engine keeps track of attributes such as the player's location, her health points, time of day, etc. The function Ψ (also part of the game framework) then converts this state into a textual description of the location the player is at or a message indicating low health. We do not assume the agent has access to either H or Ψ, during both training and testing phases of our experiments. The space of all possible text descriptions is denoted by S. Rewards are generated using R and provided to the player upon completion of in-game quests. Figure 2-2 provides a graphical illustration.

2.3.2 Q-Learning

Reinforcement Learning is a commonly used framework for learning control policies in game environments [101, 3, 15, 106]. The game environment can be formulated as a sequence of state transitions (s, a, r, s') of a Markov Decision Process (MDP). The agent takes an action a in state s by consulting a state-action value function $Q(s, a)$, which is a measure of the action's expected long-term reward. Q-Learning [117] is a model-free technique which is used to learn an optimal $Q(s, a)$ for the agent. Starting from a random Q-function, the agent continuously updates its Q-values by playing the game and obtaining rewards. The iterative updates are derived from the Bellman equation [105]:

$$Q_{i+1}(s, a) = \mathrm{E}[r + \gamma \max_{a'} Q_i(s', a') \mid s, a] \tag{2.1}$$

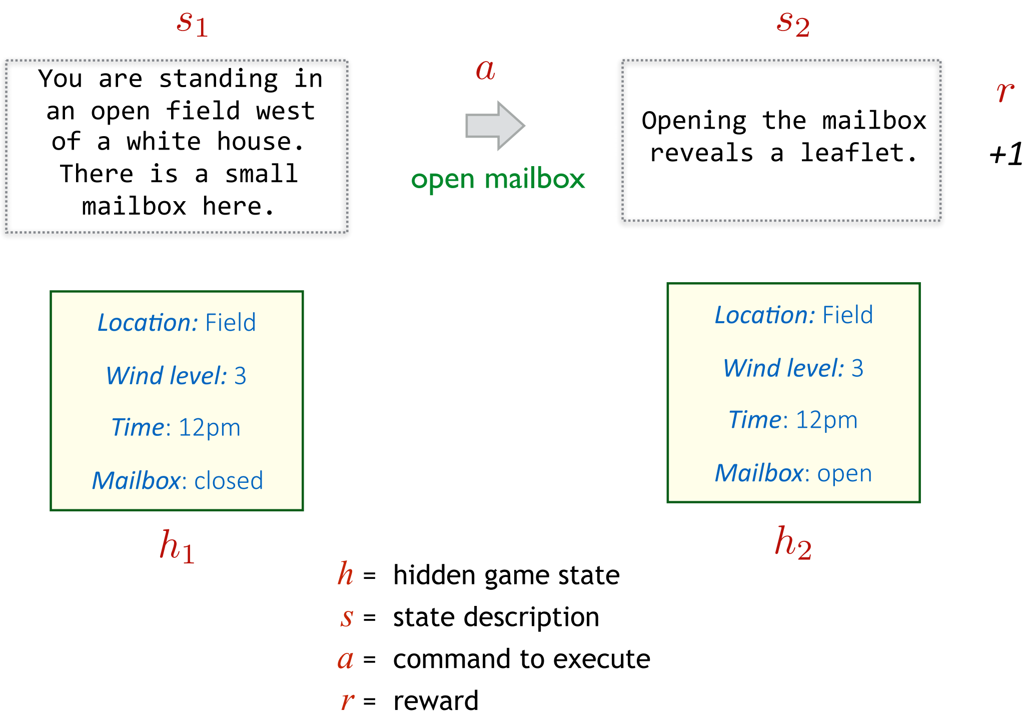


Figure 2-2: Sample transition in a text-based game. Although the game state is tracked in a structured form (h), the player only observes the state descriptions s, based on which she chooses actions a.

where γ is a discount factor for future rewards and the expectation is over all game transitions that involved the agent taking action a in state s.

Using these evolving Q-values, a reasonable policy for the agent is to choose the action with the highest $Q(s, a)$, in order to maximize its expected future rewards:

$$\pi(s) = \arg\max_a Q(s, a) \tag{2.2}$$

For effective policy learning, however, we require a trade-off between exploration of the state space and exploitation of our current policy. This can be achieved following an ϵ-greedy policy [105], where the agent performs a random action with probability ϵ, and chooses $\pi(s)$ otherwise.

2.3.3 Deep Q-Network

In large games, it is often impractical to maintain the Q-value for all possible state-action pairs. One solution to this problem is to approximate $Q(s, a)$ using a parametrized function $Q(s, a; \theta)$, which can generalize over states and actions by considering higher-level attributes [105, 14]. However, creating a good parametrization requires knowledge of the state and action spaces. One way to bypass this feature engineering is to use a Deep Q-Network (DQN) [73]. The DQN approximates the Q-value function with a deep neural network to predict $Q(s, a)$ for all possible actions a simultaneously given the current state s. The non-linear function layers of the DQN also enable it to learn better value functions than linear approximators.

2.4 Model architecture

Designing an autonomous player for text-based games provides two main challenges. First, we need a versatile representation for text that can capture the compositional aspects of natural language. Second, we require a value function approximator that can operate over this representation, and generalizing well over large state spaces. We require the model to provide good Q-value approximations for these games, even with stochastic textual descriptions.

To satisfy these needs, we divide our model (LSTM-DQN) into two parts. The first module is a *representation generator* that converts the textual description of the current state into a real-valued vector. This vector is then input into the second module which is an *action scorer*. Figure 2-3 shows the overall architecture of our model. We learn the parameters of both the representation generator and the action scorer jointly, using the in-game reward feedback. Specific implementation details are provided in Section 2.6.4.

2.4.1 Representation Generator (ϕ_R)

The representation generator reads raw text displayed to the agent and converts it to a vector representation v_s. A bag-of-words (BOW) representation is not sufficient to capture higher-order structures of sentences and paragraphs. The need for a better semantic representation of the text is evident from the average performance of this representation in playing MUD-games (as we show in Section 2.7).

In order to assimilate better representations, we utilize a Long Short-Term Memory network (LSTM) [50] as a representation generator. LSTMs are recurrent neural networks with the ability to connect and recognize long-range patterns between words in text. They are more robust than BOW to small variations in word usage and are able to capture underlying semantics of sentences to some extent. In recent work, LSTMs have been used successfully in NLP tasks such as machine translation [104] and sentiment analysis [107] to compose vector representations of sentences from

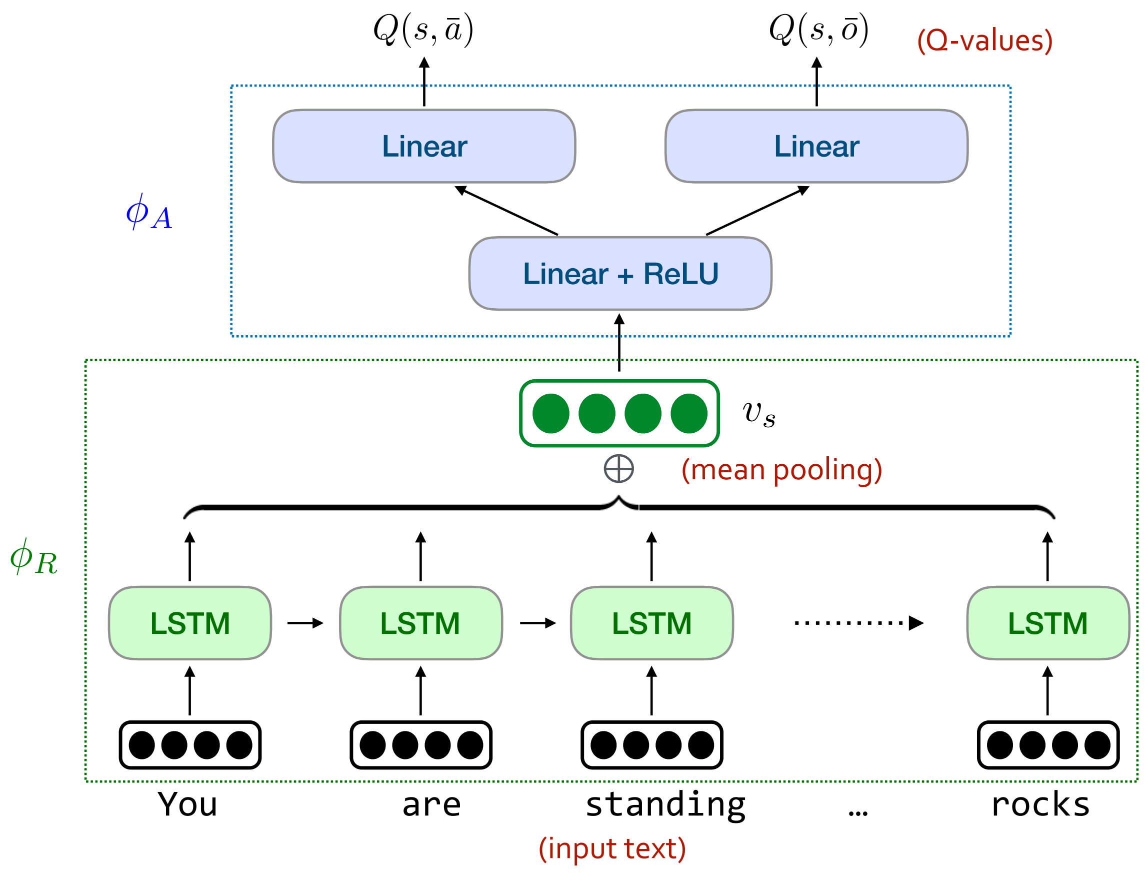


Figure 2-3: Architecture of LSTM-DQN: The Representation Generator (ϕ_R) (bottom) takes as input a stream of embeddings corresponding to words observed in state s and produces a single vector representation v_s. This vector is then fed into the action scorer (ϕ_A) (top) to produce Q-values for all actions and argument objects.

word-level embeddings [71, 87]. In our setup, the LSTM network takes in word embeddings w_k from the words in a description s and produces output vectors x_k at each step.

An LSTM cell consists of an input gate i, a forget gate f and a memory cell c. At each step k, the LSTM takes in the following vectors in $\mathbb{R}^d$: an input word embedding w_k, the output from the previous step h_{k-1} and a memory vector c_{k-1}, also from the previous step. The cell produces an output vector x_k at each step, which is an accumulated representation over the preceding words up till the current point. The transition equations for the LSTM can be summarized as:

$$\begin{aligned}
i_k &= \sigma(U^{(i)}w_k + V^{(i)}h_{k-1} + b^{(i)}), \\
f_k &= \sigma(U^{(f)}w_k + V^{(f)}h_{k-1} + b^{(f)}), \\
o_k &= \sigma(U^{(o)}w_k + V^{(o)}h_{k-1} + b^{(o)}) \\
z_k &= \tanh(U^{(z)}w_k + V^{(z)}h_{k-1} + b^{(z)}) \\
c_k &= i_k \odot z_k + f_k \odot c_{k-1} \\
x_k &= o_k \odot \tanh(c_k)
\end{aligned} \tag{2.3}$$

where σ represents the sigmoid function and $\odot$ is elementwise multiplication. To arrive at the final state representation v_s, we add a *mean pooling* layer which computes the element-wise mean over the output vectors x_k.[2]

$$v_s = \frac{1}{n}\sum_{k=1}^{n} x_k \tag{2.4}$$

2.4.2 Action Scorer (ϕ_A)

The action scorer module produces scores for the set of possible actions given the current state representation. We use a multi-layered neural network for this purpose (see Figure 2-3). The input to this module is the vector from the representation generator, $v_s = \phi_R(s)$ and the outputs are scores for actions $a \in A$. Scores for all actions are predicted simultaneously, which is computationally more efficient than scoring each state-action pair separately. Thus, by combining the representation generator and action scorer, we can obtain the approximation for the Q-function as:

$$Q(s, a) \approx \phi_A(\phi_R(s))[a] \tag{2.5}$$

An additional complexity in playing MUD games is that the actions taken by the player are multi-word natural language *commands* such as *'eat apple'* or *'go east'*. Due to computational constraints, we limit ourselves to commands consisting of a single

[2]We also experimented with considering just the output vector of the LSTM after processing the last word. Empirically, we find that mean pooling leads to faster learning, and hence use it in all our experiments.

action word (e.g. *eat*) with a single argument object (e.g. *apple*). This assumption holds for the majority of commands in our worlds, with the exception of one class that requires two arguments (e.g. *move red-root right*, *move blue-root up*). We take into account all possible actions and objects available in the game and predict both for each state using the same network (Figure 2-3). The Q-value of the entire command $(\hat{a}, \hat{o})$ is taken to be the average of the Q-values predicted for the action $\hat{a}$ and the object $\hat{o}$. For the rest of this section, we show equations for the overall $Q(s, a)$.

2.5 Parameter Learning

The two components of our model fit together to form a single unit that can be trained end-to-end. We jointly learn the parameters θ_R of both the representation generator and θ_A of the action scorer using stochastic gradient descent with *RMSprop* [111]. The complete training procedure is shown in Algorithm 2.1. In each iteration i, we update the parameters to reduce the discrepancy between the predicted value of the current state $Q(s_t, a_t; \theta_i)$ (where $\theta_i = [\theta_R; \theta_A]_i$) and the expected Q-value given the reward r_t and the value of the next state $\max_a Q(s_{t+1}, a; \theta_{i-1})$.

We keep track of the agent's previous experiences in a *memory* $\mathcal{D}$.[3] Instead of performing updates to the Q-value using transitions from the current episode, we sample a random transition $(\hat{s}, \hat{a}, s', r)$ from $\mathcal{D}$. Updating the parameters in this way avoids issues due to strong correlation when using transitions of the same episode [73]. Using the sampled transition and (2.1), we obtain the following loss function to minimize:

$$\mathcal{L}_i(\theta_i) = \mathrm{E}_{\hat{s},\hat{a}}[(Q(\hat{s}, \hat{a}; \theta_i) - y_i)^2] \tag{2.6}$$

where $y_i = \mathrm{E}_{\hat{s},\hat{a}}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) \mid \hat{s}, \hat{a}]$ is the target Q-value with parameters θ_{i-1} fixed from the previous iteration.

The updates on the parameters θ can be performed using the following gradient of $\mathcal{L}_i(\theta_i)$:

$$\nabla_{\theta_i} \mathcal{L}_i(\theta_i) = \mathrm{E}_{\hat{s},\hat{a}}[2(Q(\hat{s}, \hat{a}; \theta_i) - y_i) \nabla_{\theta_i} Q(\hat{s}, \hat{a}; \theta_i)] \tag{2.7}$$

For each epoch of training, the agent plays several episodes of the game, which is restarted after every terminal state. We perform linear annealing of both ϵ and the learning rate as training progresses.

Mini-batch Sampling In practice, online updates to the parameters θ are performed over a mini batch of state transitions, instead of a single transition. This increases the number of experiences used per update step and is also more efficient due to optimized matrix operations.

[3]The memory is limited and rewritten in a first-in-first-out (FIFO) fashion.

Algorithm 2.1 Training Procedure for LSTM-DQN with prioritized sampling

```
1: Initialize experience memory $\mathcal{D}$
2: Initialize parameters of representation generator ($\phi_R$) and action scorer ($\phi_A$) randomly
3: for $episode = 1, M$ do
4:     Initialize game and get start state description $s_1$
5:     for $t = 1, T$ do
6:         if $random() < \epsilon$ then
7:             Select a random action $a_t$
8:         else
9:             Convert text $s_t$ to representation $v_{s_t}$ using $\phi_R$
10:            Compute $Q(s_t, a)$ for all actions using $\phi_A(v_{s_t})$
11:            Select $a_t = \text{argmax}\ Q(s_t, a)$
12:        Execute action $a_t$ and observe reward $r_t$ and new state $s_{t+1}$
13:        Set priority $p_t = 1$ if $r_t > 0$, else $p_t = 0$
14:        Store transition $(s_t, a_t, r_t, s_{t+1}, p_t)$ in $\mathcal{D}$
15:        Sample random mini batch of transitions $(s_j, a_j, r_j, s_{j+1}, p_j)$ from $\mathcal{D}$,
             with fraction $\rho$ having $p_j = 1$
16:        Set $y_j = \begin{cases} r_j & \text{if } s_{j+1} \text{ is terminal} \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta) & \text{if } s_{j+1} \text{ is non-terminal} \end{cases}$
17:        Perform gradient descent step on the loss $\mathcal{L}(\theta) = (y_j - Q(s_j, a_j; \theta))^2$
```

The simplest method to create these mini-batches from the experience memory $\mathcal{D}$ is to sample uniformly at random. However, certain experiences are more valuable than others for the agent to learn from. For instance, rare transitions that provide positive rewards can be used more often to learn optimal Q-values faster. In our experiments, we consider such positive-reward transitions to have higher *priority* and keep track of them in $\mathcal{D}$. We perform *prioritized sampling* (inspired by prioritized sweeping [74]) to sample a fraction ρ of transitions from the higher priority pool and a fraction $1 - \rho$ from the rest.

2.6 Experimental Setup

2.6.1 Game Environment

For our game environment, we modify Evennia,[4] an open-source library for building online textual MUD games. Evennia is a Python-based framework that allows one to easily create new games by writing a batch file describing the environment with details of rooms, objects and actions. The game engine keeps track of the game state internally, presenting textual descriptions to the player and receiving text commands from the player. We conduct experiments on two worlds - a smaller *Home world* we created ourselves, and a larger, more complex *Fantasy world* created by Evennia's developers. The motivation behind Home world is to abstract away high-level planning and focus on the language understanding requirements of the game.

Table 2.1 provides statistics of the narratives in the game worlds. We observe that the Fantasy world is moderately sized with a vocabulary of 1340 words and up to 100 different descriptions for a room. These descriptions were created manually by the game developers. These diverse, engaging descriptions are designed to make it interesting and exciting for human players. Several rooms have many alternative descriptions, invoked randomly on each visit by the player.

Comparatively, the Home world is smaller: it has a restricted vocabulary of 84 words and the room descriptions are relatively structured. However, both the room descriptions (which are also varied and randomly provided to the agent) and the quest descriptions were adversarially created with linguistic concepts such as negation and conjunction to force an agent to properly understand the state in order to play well. Therefore, this domain provides an interesting challenge for language understanding.

Rewards In both worlds, the agent receives a positive reward on completing a quest, and negative rewards for getting into bad situations like falling off a bridge, or losing a fight. We also add small deterministic negative rewards for each non-terminating

[4] `http://www.evennia.com/`

Stats	**Home World**	**Fantasy World**
Vocabulary size	84	1340
Avg. words / description	10.5	65.21
Max descriptions / room	3	100
# diff. quest descriptions	12	-
State transitions	Deterministic	Stochastic
# states (underlying)	16	≥ 56
Branching factor (# commands / state)	40	222

Table 2.1: Various statistics for the two game worlds.

step. This incentivizes the agent to learn policies that solve quests in fewer steps. Details on reward structure are provided in Appendix A.

Home World We created *Home world* to mimic the environment of a typical house.[5] The world consists of four rooms - a living room, a bedroom, a kitchen and a garden with connecting pathways. Every room is reachable from every other room. Each room contains a representative object that the agent can interact with. For instance, the kitchen has an *apple* that the player can *eat*. Transitions between the rooms are deterministic. At the start of each game episode, the player is placed in a random room and provided with a randomly selected quest. The text provided to the player contains both the description of her current state and that of the quest. Thus, the player can begin in one of 16 different states (4 rooms $\times$ 4 quests), which adds to the world's complexity.

An example of a quest given to the player in text is *'Not you are sleepy now but you are hungry now'*. To complete this quest and obtain a reward, the player has to navigate through the house to reach the kitchen and eat the apple (i.e type in the command *eat apple*). More importantly, the player should interpret that the quest does not require her to take a nap in the bedroom. We created such misguiding quest prompts to make it hard for agents to succeed without having an adequate level of language understanding.

[5]An illustration is provided in Appendix A.

Fantasy World The Fantasy world is considerably more complex and involves quests such as navigating through a broken bridge or finding the secret tomb of an ancient hero. This game also has stochastic transitions in addition to varying state descriptions provided to the player. For instance, there is a possibility of the player falling from the bridge if she lingers too long on it.

Due to the large command space in this game,[6] we make use of cues provided by the game itself to narrow down the set of possible objects to consider in each state. For instance, in the MUD example in Figure 1, the game provides a list of possible exits. If the game does not provide such clues for the current state, we consider all objects in the game.

2.6.2 Evaluation

We use two metrics for measuring an agent's performance: (1) the average cumulative reward obtained per episode, and (2) the fraction of quests completed by the agent. Although these measures are correlated to an extent, there do exist scenarios when some agents do well on one and poorly on the other (as we demonstrate in our experiments). In a single epoch, we first train the agent on M episodes of T steps each. At the end of this training, we have a testing phase of running M episodes of the game, again for T steps each. We use $M = 50, T = 20$ for the Home world and $M = 20, T = 250$ for the Fantasy world. For all evaluation episodes, we run the agent following an ϵ-greedy policy with $\epsilon = 0.05$, which makes the agent choose the best action according to its Q-values 95% of the time.

2.6.3 Baselines

We compare our model (LSTM-DQN) with three baselines. The first is a *Random* agent that chooses both actions and objects uniformly at random from all available choices.[7] This baseline helps us test whether the game world is easy enough to solve

[6]We consider 222 possible command combinations of 6 actions and 37 object arguments.

[7]In the case of the Fantasy world, the object choices are narrowed down using game clues as described earlier.

by performing random moves. The other two baselines are BOW-DQN and BI-DQN, which use a bag-of-words and a bag-of-bigrams representation of the text as input to a DQN action scorer. These baselines serve to illustrate the importance of having a good representation layer for the task.

2.6.4 Implementation details

For our DQN models, we used $\mathcal{D} = 100000, \gamma = 0.5$. We use a learning rate of 0.0005 for RMSprop. We anneal the ϵ for ϵ-greedy from 1 to 0.2 over 100000 transitions. A mini-batch gradient update is performed every 4 steps of the gameplay. We roll out the LSTM (over words) for a maximum of 30 steps on the Home world and for 100 steps on the Fantasy world. For the prioritized sampling, we used $\rho = 0.25$ for both worlds. We employed a mini-batch size of 64 and word embedding size $d = 20$ in all experiments.

2.7 Results

2.7.1 Home World

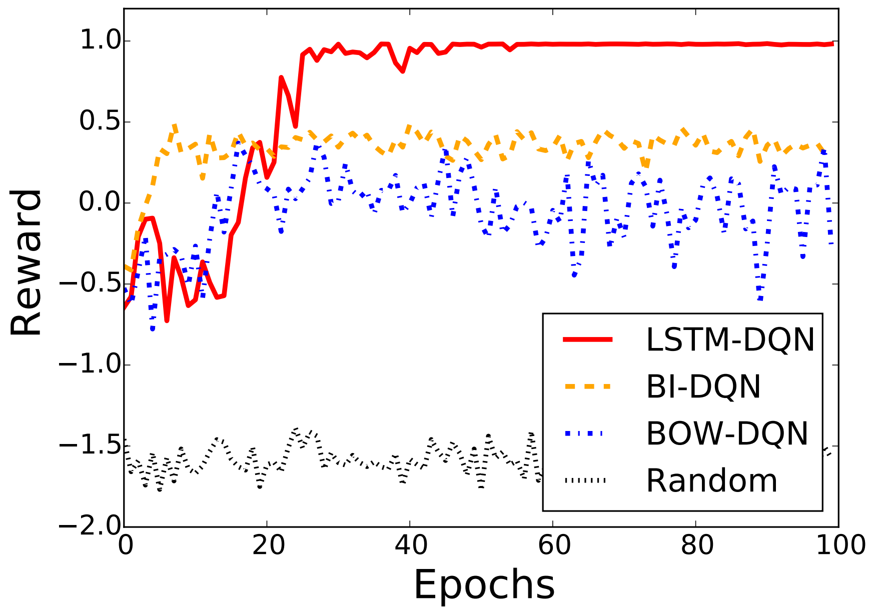
Figure 2-4 illustrates the performance of LSTM-DQN compared to the baselines. We can observe that the Random baseline performs quite poorly, completing only around 10% of quests on average[8] obtaining a low reward of around -1.58. The BOW-DQN model performs significantly better and is able to complete around 46% of the quests, with an average reward of 0.20. The improvement in reward is due to both greater quest success rate and a lower rate of issuing invalid commands (e.g. *eat apple* would be invalid in the bedroom since there is no apple). We notice that both the reward and quest completion graphs of this model are volatile. This is because the model fails to pick out differences between quests like *Not you are hungry now but you are sleepy now* and *Not you are sleepy now but you are hungry now.* The BI-DQN model suffers from the same issue although it performs slightly better than BOW-DQN by completing 48% of quests. In contrast, the LSTM-DQN model does not suffer from this issue and is able to complete 100% of the quests after around 50 epochs of training, achieving close to the optimal reward possible.[9] This demonstrates that having an expressive representation for text is crucial to understanding the game states and choosing intelligent actions.

2.7.2 Fantasy World

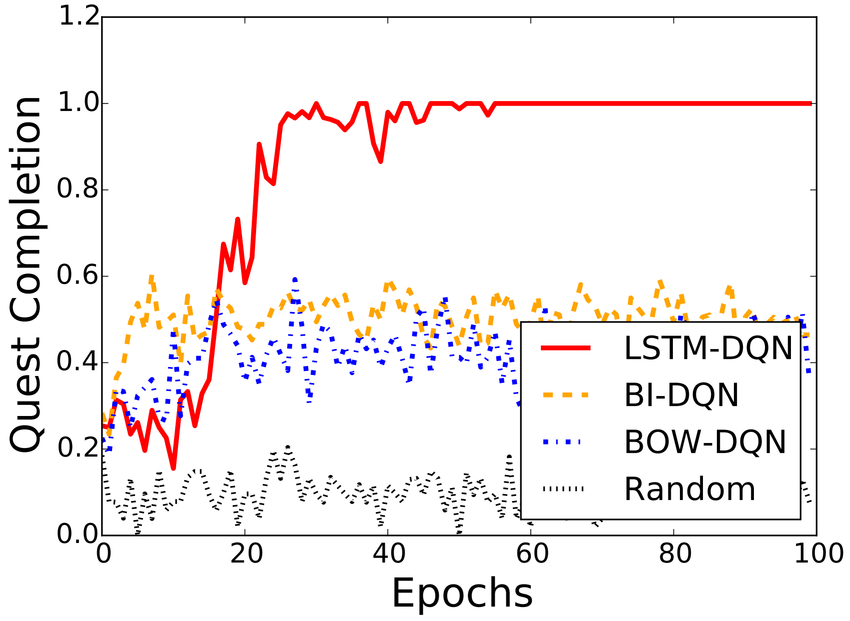
We evaluate all the models on the Fantasy world in the same manner as before and report reward, quest completion rates and Q-values. The quest we evaluate on involves crossing the broken bridge (which takes a minimum of five steps), with the possibility of falling off at random (a 5% chance) when the player is on the bridge. The game has an additional quest of reaching a secret tomb. However, this is a complex quest that requires the player to memorize game events and perform high-level planning

[8]Averaged over the last 10 epochs.

[9]Note that since each step incurs a penalty of -0.01, the best reward (on average) a player can get is around 0.98.



Reward (Home)



Quest completion (Home)

Figure 2-4: Evolution of average reward (**top**) and quest completion rate (**bottom**) for various baselines and our model (LSTM-DQN) on the Home world.

which are beyond the scope of this current work. Therefore, we focus only on the first quest.

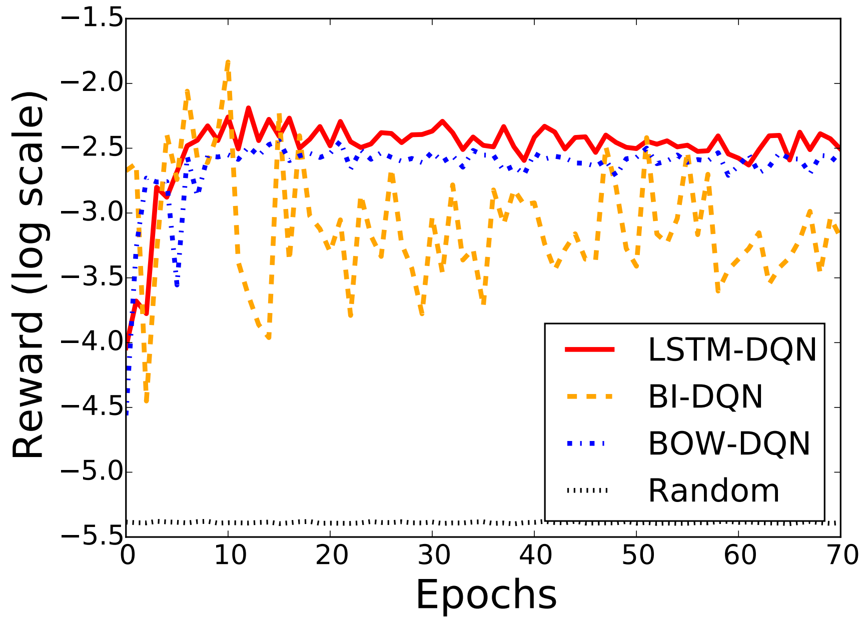
From Figure 2-5, we can see that the Random baseline does poorly in terms of both average per-episode reward[10] and quest completion rates. BOW-DQN converges to a much higher average reward of -12.68 and achieves around 82% quest completion. Again, the BOW-DQN is often confused by varying (10 different) descriptions of the portions of the bridge, which reflects in its erratic performance on the quest. The BI-DQN performs very well on quest completion by finishing 97% of quests. However, this model tends to find sub-optimal solutions and gets an average reward of -26.68, even worse than BOW-DQN. One reason for this is the negative rewards the agent obtains after falling off the bridge. The LSTM-DQN model again performs best, achieving an average reward of -11.33 and completing 96% of quests on average. Though this world does not contain descriptions adversarial to BOW-DQN or BI-DQN, the LSTM-DQN obtains higher average reward by completing the quest in fewer steps and showing more resilience to variations in the state descriptions.

2.7.3 Analysis

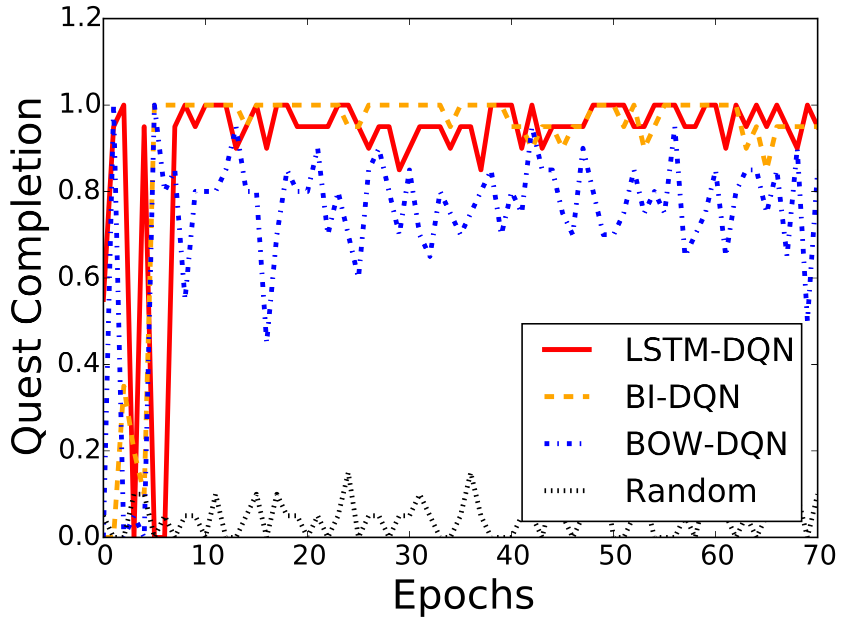
Variations of action scorer How important is the choice of the action scorer to model performance? To answer this question, we investigated the impact of using a deep neural network vs a linear model as the action scorer ϕ_A. Figure 2-6 illustrates the performance of the BOW-DQN and BI-DQN models along with their simpler versions BOW-LIN and BI-LIN, which use a single linear layer for ϕ_A. It can be seen that the DQN models clearly achieve better performance than their linear counterparts, in both the bag-of-words and bag-of-bigrams cases. This shows that a deeper neural network is capable of modeling the control policy better.

Transfer Learning We would like the representations learnt by ϕ_R to be general-purpose and *transferable* to new game worlds. To test this, we created a second Home world with the same rooms, but a completely different map, changing the locations

[10]Note that the rewards graph is in log scale.



Reward (Fantasy)



Quest completion (Fantasy)

Figure 2-5: Evolution of average reward (**top**) and quest completion rate (**bottom**) for various baselines and our model (LSTM-DQN) on the Fantasy world. Reward here is shown in log scale.

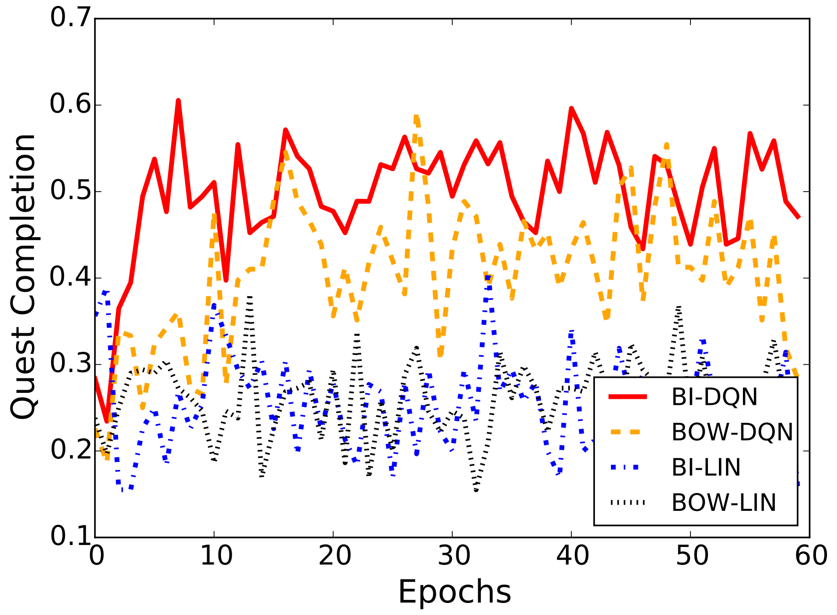


Figure 2-6: Quest completion rates of DQN vs. Linear models on Home world. The graph clearly demonstrates the advantage of using deeper models to approximate the action value function.

of the rooms and the pathways between them. The main differentiating factor of this world from the original home world lies in the high-level control policy required to complete quests.

We initialized the LSTM part of an LSTM-DQN agent with parameters θ_R learnt from the original home world and trained it on the new world.[11] Figure 2-7 demonstrates that the agent with transferred parameters is able to learn quicker than an agent starting from scratch initialized with random parameters (*No Transfer*), reaching the optimal policy almost 20 epochs earlier. This indicates that these simulated worlds can be used to learn good representations for language that transfer across worlds.

Prioritized sampling As described in Section 2.4, we employ a prioritized sampling scheme to obtain mini-batches of transitions from the experience replay. To demonstrate its usefulness, we compared the effect of different mini-batch sampling

[11]The parameters for the Action Scorer (θ_A) are initialized randomly.

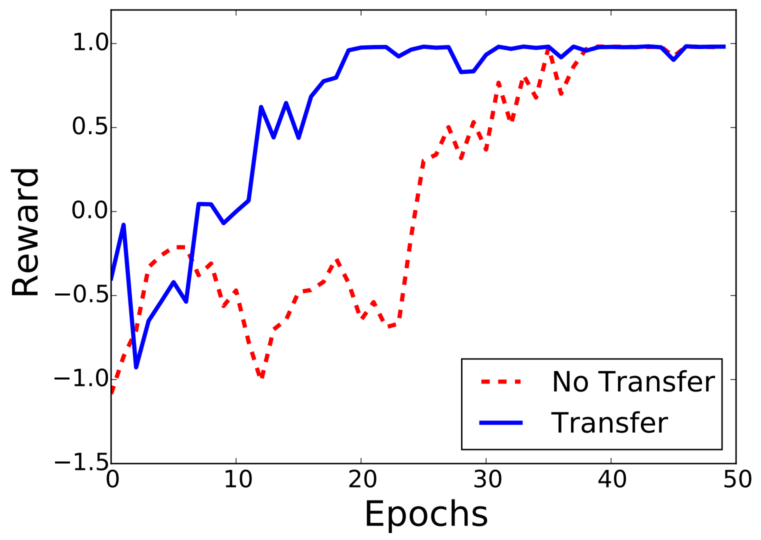


Figure 2-7: Transfer learning in the Home world. For the *Transfer* condition, a model was trained on a different world configuration (with same vocabulary), and the learned parameters of the representation generator (ϕ_R) are used in initializing the new model.

procedures on the parameter learning. From Figure 2-8, we observe that using prioritized sampling significantly speeds up learning, with the agent achieving the optimal policy around 50 epochs faster than using uniform sampling. This shows promise for further research into different schemes of assigning priority to transitions in the memory.

Representation Analysis We analyzed the representations learnt by the LSTM-DQN model on the Home world. Figure 2-9 shows a visualization of learnt word embeddings, reduced to two dimensions using t-SNE [113]. All the vectors were initialized randomly before training, and were induced using only the in-game rewards. We can see that semantically similar words appear close together to form coherent subspaces. In fact, we observe four different subspaces, each for one type of room along with its corresponding object(s) and quest words. For instance, food items like *pizza* and rooms like *kitchen* are very close to the word *hungry* which appears in a quest description. This shows that the agent learns to form meaningful associations between the semantics of the quest and the environment. Table 2.2 shows some

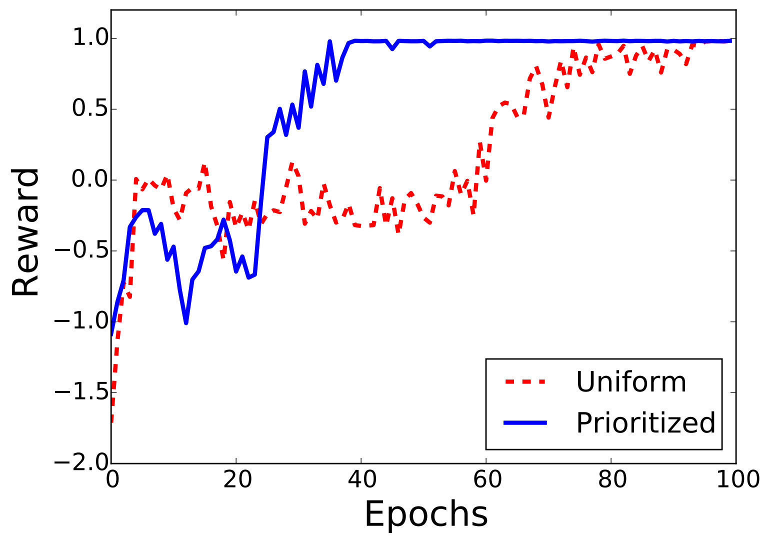


Figure 2-8: Effect of prioritized sampling on learning in the Home world. Prioritized sampling leads to faster convergence to an optimal control policy.

Description	Nearest neighbor
You are halfways out on the unstable bridge. From the castle you hear a distant howling sound, like that of a large dog or other beast.	The bridge slopes precariously where it extends westwards towards the lowest point - the center point of the hang bridge. You clasp the ropes firmly as the bridge sways and creaks under you.
The ruins opens up to the sky in a small open area, lined by columns. ... To the west is the gatehouse and entrance to the castle, whereas southwards the columns make way for a wide open courtyard.	The old gatehouse is near collapse. .... East 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.

Table 2.2: Sample descriptions from the Fantasy world and their nearest neighbors according to their vector representations from the LSTM representation generator. The NNs are often descriptions of the same or similar (nearby) states in the game.

examples of descriptions from Fantasy world and their nearest neighbors using cosine similarity between their corresponding vector representations produced by LSTM-DQN. The model is able to correlate descriptions of the same (or similar) underlying states and project them onto nearby points in the representation subspace.

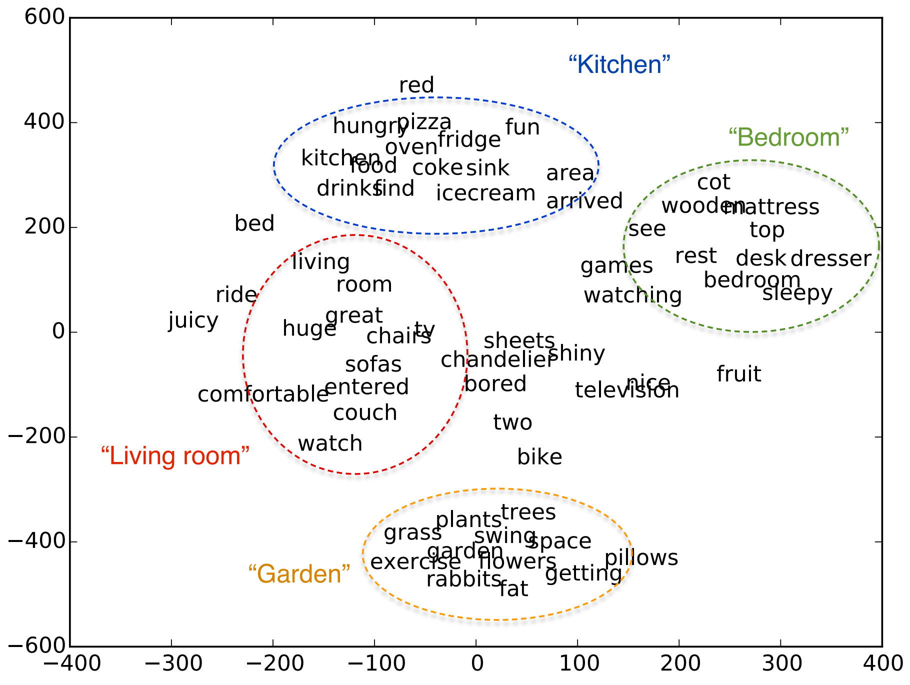


Figure 2-9: t-SNE visualization of word embeddings (except stopwords) after training on Home world. Note that these embedding values are initialized to random and learned using only in-game rewards.

2.8 Conclusions

In this chapter, we presented a model for end-to-end learning of control policies for text-based games. Using game rewards as feedback, we demonstrated that optimized representations for natural language can be learned and employed successfully to inform an effective control policy. From experiments on two domains, we show that our model outperforms several baselines that use less expressive representations for the text. We also discover 'meaningful' patterns in the representations learned by the model (both for words and sentences). Below, we provide an account of some follow-up work that has been carried out hence, and describe some future directions of investigation.

Future directions

- One limitation of our model is that we restrict ourselves to choosing two-word commands. Recent work by He et al. [46] has relaxed this assumption to include longer text responses from the player to the game engine. However, their model makes a choice between pre-defined responses, by learning a representation for each response and scoring it conditioned on the current state description. An interesting direction to pursue is models that can learn to generate phrase-level responses (say, using a sequence-level decoder [104]).

- Another element lacking in our model is long-term planning. Having a memory component can help an agent remember pieces of information from previous states (more than a few time steps in the past), which is often crucial for navigating these environments. The integration of modules like memory networks [119] or hierarchical deep Q-networks [60] could lead to successfully solving more challenging puzzles in these games.

- Improving the sampling scheme for mini-batches from the experience replay is another area for future investigation. Recently, Schaul et al. [98] proposed a more general version of our prioritized sampling scheme, demonstrating consistent improvements to learning. Developing more efficient schemes that can pick out diverse experiences are bound to provide a boost to learning.

Chapter 3

Policy Transfer via Language Grounding

This chapter presents a method for grounding natural language to adapt and transfer policies across domains. Specifically, we demonstrate that textual descriptions of environments provide a compact intermediate channel that facilitates effective policy transfer. We employ a model-based reinforcement learning approach consisting of a value iteration network, a model-free policy estimator and a two-part representation to effectively utilize entity descriptions. We demonstrate that grounding text semantics to the dynamics of the environment significantly outperforms other approaches on both transfer and multi-task scenarios in a variety of different environments.

3.1 Introduction

Deep reinforcement learning has emerged as a method of choice for many control applications ranging from computer games [73, 100] to robotics [64]. However, the success of this approach depends on a substantial number of interactions with the environment during training, easily reaching millions of steps [76, 72]. Moreover, given a new task, even a related one, this training process has to be performed from scratch. This inefficiency has motivated recent work in learning universal policies that can generalize across related tasks [97], as well as other transfer approaches [85, 91, 31, 112, 92, 93]. In this paper, we explore transfer methods that utilize text descriptions to facilitate policy generalization across tasks.

As an example, consider the game environments in Figure 3-1. The two games – *Boulderchase* and *Bomberman* – differ in their layouts and entity types. However, the high-level behavior of most entities in both games is similar. For instance, the *scorpion* in Boulderchase (left) is a moving entity which the agent has to avoid, similar to the *spider* in Bomberman (right). Though the similarity is clearly reflected in the text descriptions in Figure 3-1, it may take multiple environment interactions to discover this. Therefore, exploiting these textual clues could help an autonomous agent understand this connection more effectively, leading to faster policy learning.

To test this hypothesis, we consider multiple environments augmented with textual descriptions. These descriptions provide a short overview of the objects and their modes of interaction in the environment. They do not describe control strategies, which were commonly used in prior work on grounding [114, 14]. Instead, they specify the dynamics of the environments, which are more conducive to cross-domain transfer.

In order to effectively utilize this type of information, we employ a model-based reinforcement learning approach. Typically, representations of the environment learned by these approaches are inherently domain specific. We address this issue by using natural language as an implicit intermediate channel for transfer. Specifically, our model learns to map text descriptions to transitions and rewards in an environment, a capability that speeds up learning in unseen domains. We induce a two-part representation for the input state that generalizes over domains, incorporating both domain-specific information and textual knowledge. This representation is utilized by an action-value function, parametrized as a single deep neural network with a differentiable value iteration module [108]. The entire model is trained end-to-end using rewards from the environment.

We evaluate our model on several game worlds from the GVGAI framework [88]. In our first evaluation scenario of transfer learning, an agent is trained on a set of

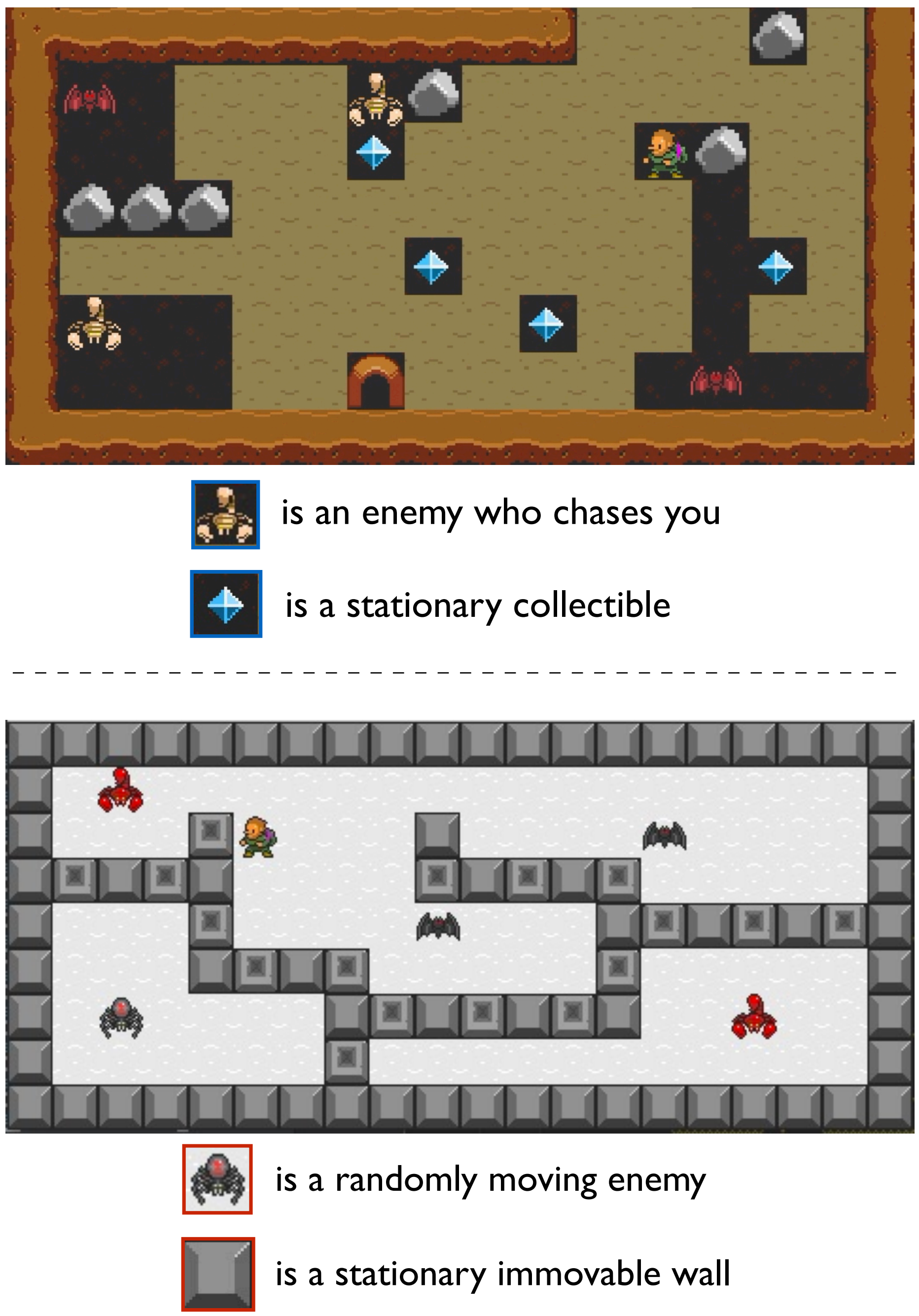

is an enemy who chases you
is a stationary collectible
is a randomly moving enemy
is a stationary immovable wall

Figure 3-1: Examples of two different game environments, Boulderchase **(top)** and Bomberman **(bottom)**. Each domain also has text descriptions associated with specific entities, describing characteristics such as movement and interactions with the player's avatar.

source tasks and its learning performance is evaluated on a different set of target tasks. Across multiple evaluation metrics, our method consistently outperforms several baselines and an existing transfer approach called Actor Mimic [85]. For instance, our model achieves up to 35% higher average reward and up to 15% higher jumpstart reward. We also evaluate on a multi-task setting where learning is simultaneously performed on multiple environments. In this case, we obtain gains of up to 30% and 7% on average and asymptotic reward, respectively.

The remainder of this chapter is organized as follows. Section 3.2 summarized related work on grounding and transfer for reinforcement learning; Section 3.3 provides an overview of the framework we use; Section 3.4 describes our model architecture and its various components; Section 3.5 details the experimental setup, and Section 3.6 contains our empirical results and analysis. We conclude and discuss some future directions in Section 3.7.

3.2 Related Work

3.2.1 Grounding language in interactive environments

In recent years, there has been increasing interest in systems that can utilize textual knowledge to learn control policies. Such applications include interpreting help documentation [16], instruction following [114, 56, 6, 69, 5] and learning to play computer games [14, 78]. In all these applications, the models are trained and tested on the same domain.

Our work represents two departures from prior work on grounding. First, rather than optimizing control performance for a single domain, we are interested in the multi-domain transfer scenario, where language descriptions drive generalization. Second, prior work uses text in the form of strategy advice to directly learn the policy. Since the policies are typically optimized for a specific task, they may be harder to transfer across domains. Instead, we utilize text to bootstrap the induction of the environment dynamics, moving beyond task-specific strategies.

Another related line of work consists of systems that learn spatial and topographical maps of the environment for robot navigation using natural language descriptions [115, 47]. These approaches use text mainly containing appearance and positional information, and integrate it with other semantic sources (such as appearance models) to obtain more accurate maps. In contrast, our work uses language describing the dynamics of the environment, such as entity movements and interactions, which is complementary to positional information received through state observations. Further, our goal is to help an agent learn policies that generalize over different stochastic domains, while their work considers a single domain.

3.2.2 Transfer in Reinforcement Learning

Transferring policies across domains is a challenging problem in reinforcement learning [58, 110]. The main hurdle lies in learning a good mapping between the state and action spaces of different domains to enable effective transfer. Most previous approaches have either explored skill transfer [57] or value function transfer [66]. There have been a few attempts at model-based transfer for RL [109, 81, 37, 116] but these methods either rely on hand-coded inter-task mappings for state and actions spaces or require significant interactions in the target task to learn an effective mapping. Our approach doesn't use any explicit mappings and can learn to predict the dynamics of a target task using its descriptions.

A closely related line of work concerns transfer methods for deep reinforcement learning. Parisotto et al. [85] train a deep network to mimic pre-trained experts on source tasks using policy distillation. The learned parameters are then used to initialize a network on a target task to perform transfer. Rusu et al. [92] perform transfer by freezing parameters learned on source tasks and adding a new set of parameters for every new target task, while using both sets to learn the new policy. Work by Rajendran et al. [91] uses attention networks to selectively transfer from a set of expert policies to a new task. Our approach is orthogonal since we use text to bootstrap transfer, and can potentially be combined with these methods to achieve more effective transfer.

Perhaps closest in spirit to our hypothesis is the recent work of Harrison et al. [45]. Their approach makes use of paired instances of text descriptions and state-action information from human gameplay to learn a machine translation model. This model is incorporated into a policy shaping algorithm to better guide agent exploration. Although the motivation of language-guided control policies is similar to ours, their work considers transfer across tasks in a single domain, and requires human demonstrations to learn a policy.

3.3 General Framework

3.3.1 Environment Setup

We model a single environment as a Markov Decision Process (MDP), represented by $E = \langle S, A, T, R, O, Z \rangle$. Here, S is the state space, and A is the set of actions available to the agent. In this work, we consider every state $s \in S$ to be a 2-dimensional grid of size $m \times n$, with each cell containing an entity symbol $o \in O$.[1] T is the transition distribution over all possible next states s' conditioned on the agent choosing action a in state s. R determines the reward provided to the agent at each time step. The agent does not have access to the true T and R of the environment. Each domain also has a goal state $s_g \in S$ which determines when an episode terminates. Finally, Z is the complete set of text descriptions provided to the agent for this particular environment.

3.3.2 Reinforcement learning (RL)

The goal of an autonomous agent is to maximize cumulative reward obtained from the environment. A traditional way to achieve this is by learning an action value function $Q(s, a)$ through reinforcement. The *Q-function* predicts the expected future reward for choosing action a in state s. A straightforward policy then is to simply choose the action that maximizes the Q-value in the current state: $\pi(s) = \arg\max_a Q(s, a)$. If we also make use of the descriptions, we have a text-conditioned policy: $\pi(s, Z) = \arg\max_a Q(s, a, Z)$.

A successful control policy for an environment will contain both knowledge of the environment dynamics and the capability to identify goal states. While the latter is task-specific, the former characteristic is more useful for learning a general policy that transfers to different domains. Based on this hypothesis, we employ a model-aware RL approach that can learn the dynamics of the world while estimating the optimal

[1]In our experiments, we relax this assumption to allow for multiple entities per cell, but for ease of description, we shall assume a single entity. The assumption of 2-D worlds can also be easily relaxed to generalize our model to other situations.

Q. Specifically, we make use of *Value Iteration (VI)* [105], an algorithm based on dynamic programming. The update equations are as follows:

$$\begin{aligned} Q^{(n+1)}(s,a,Z) &= R(s,a,Z) \\ &\quad + \gamma \sum_{s' \in S} T(s'|s,a,Z) V^{(n)}(s',Z) \\ V^{(n+1)}(s,Z) &= \max_a Q^{(n+1)}(s,a,Z) \end{aligned} \tag{3.1}$$

where γ is a discount factor and n is the iteration number. The updates require an estimate of T and R, which the agent must obtain through exploration of the environment.

3.3.3 Text descriptions

Estimating the dynamics of the environment from interactive experience can require a significant number of samples. Our main hypothesis is that if an agent can derive information about the dynamics from text descriptions, it can determine T and R faster and more accurately.

For instance, consider the sentence *"Red bat that moves horizontally, left to right."*. This talks about the movement of a third-party entity ('bat'), independent of the agent's goal. Provided the agent can learn to interpret this sentence, it can then infer the direction of movement of a different entity (e.g. *"A tan car moving slowly to the left"* in a different domain. Further, this inference is useful even if the agent has a completely different goal. On the other hand, instruction-like text, such as *"Move towards the wooden door"*, is highly context-specific, only relevant to domains that have the mentioned goal.

With this in mind, we provide the agent with text descriptions that collectively portray characteristics of the world. A single description talks about one particular entity in the world. The text contains (partial) information about the entity's movement and interaction with the player avatar. Each description is also aligned to its corresponding entity in the environment. Figure 3-2 provides some samples; details

- Scorpion2: *Red scorpion that moves up and down*
- Alien3: *This character slowly moves from right to left while having the ability to shoot upwards*
- Sword1: *This item is picked up and used by the player for attacking enemies*

Figure 3-2: Example text descriptions of entities in different environments, collected using Amazon Mechanical Turk.

on data collection and statistics are in Section 3.5. More descriptions are provided in Appendix B.

3.3.4 Transfer for RL

A natural scenario to test our grounding hypothesis is to consider learning across multiple environments. The agent can learn to ground language semantics in an environment E_1 and then we can test its understanding capability by placing it in a new unseen domain, E_2. The agent is allowed unlimited experience in E_1, and after convergence of its policy, it is then allowed to interact with and learn a policy for E_2. We do not provide the agent with any mapping between entities or goals across domains, either directly or through the text. The agent's goal is to re-utilize information obtained in E_1 to learn more efficiently in E_2.

3.4 Model

Grounding language for policy transfer across domains requires a model that meets two needs. First, it must allow for a flexible representation that fuses information from both state observations and text descriptions. This representation should capture the compositional nature of language while mapping linguistic semantics to characteristics of the world. Second, the model must have the capability to learn an accurate prototype of the environment (i.e. transitions and rewards) using only interactive feedback. Overall, the model must enable an agent to map text descriptions to environment dynamics; this allows it to predict transitions and rewards in a completely new world, without requiring substantial interaction.

To this end, we propose a neural architecture consisting of two main components: (1) a *representation generator* (ϕ), and (2) a *value iteration network* (VIN) [108]. The representation generator takes the state observation and the set of text descriptions to produce a tensor, capturing essential information for decision making. The VIN module implicitly encodes the value iteration computation (Eq. 3.1) into a recurrent network with convolutional modules, producing an action-value function using the tensor representation as input. Together, both modules form an end-to-end differentiable network that can be trained using simple back-propagation.

3.4.1 Representation generator

The main purpose of this module is to fuse together information from two inputs - the state, and the text specifications. An important consideration, however, is the ability to handle partial or incomplete text descriptions, which may not contain all the particulars on an entity. Thus, we would like to incorporate useful information from the text, yet, not rely on it completely. This motivates us to utilize a factorized representation over the two input modalities.

Formally, given a state matrix s and a set of text descriptions Z, the module produces a tensor $\phi(s, Z)$. Consider a cell in s containing an entity o_i, with a cor-

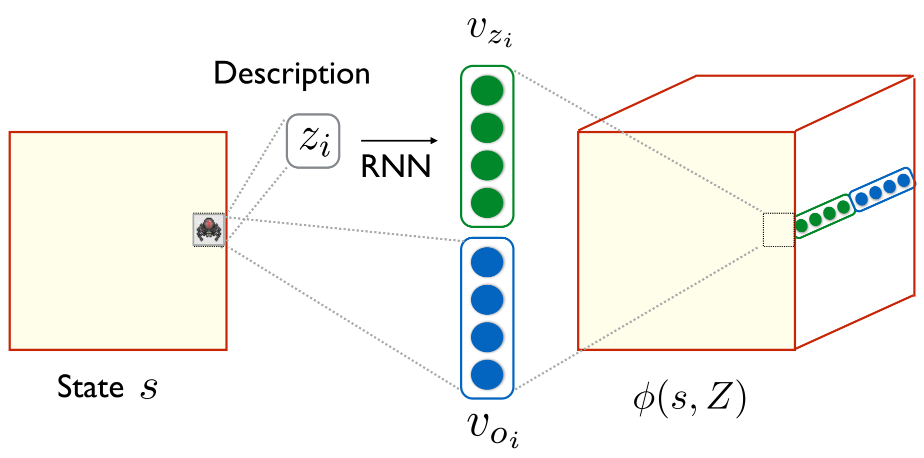


Figure 3-3: Representation generator combining both object-specific and description-informed vectors for each entity. Each cell in the input state (2-D matrix) is converted to a corresponding real-valued vector, resulting in a 3-D tensor output.

responding description z_i (if available). Each such cell is converted into a vector $\phi_i = [v_{o_i}; v_{z_i}]$, consisting of two parts concatenated together:

1. v_{o_i}, which is an entity-specific vector embedding of dimension d
2. v_{z_i} (also of dimension d), produced from z_i using an LSTM recurrent neural network [50].

This gives us a tensor ϕ with dimensions $m \times n \times 2d$ for the entire state. For cells with no entity (i.e. empty space), ϕ_i is simply a zero vector, and for entities without a description, $v_{z_i} = \vec{0}$. Figure 3-3 illustrates this module.

This decomposition allows us to learn policies based on both the ID of an object and its described behavior in text. In a new environment, previously seen entities can reuse the learned representations directly based on their symbols. For completely new entities (with unseen IDs), the model can form useful representations from their text descriptions.

3.4.2 Value iteration network

For a model-based RL approach to this task, we require some means to estimate T and R of an environment. One way to achieve this is by explicitly using predictive

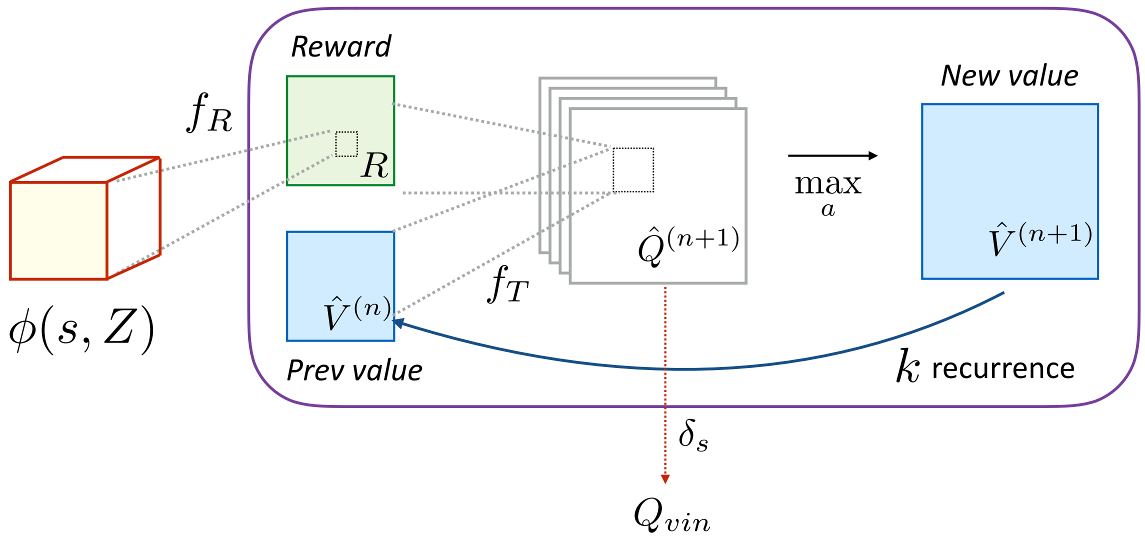


Figure 3-4: Value iteration network (VIN) module to compute Q_{vin} from $\phi(s, Z)$. The module approximates the value iteration computation using neural networks to predict reward and value maps, arranged in a recurrent fashion. Functions f_T and f_R are implemented using convolutional neural networks (CNNs). δ_s is a selection function to pick out a single Q-value (at the agent's current location) from the output Q-value map $\hat{Q}^{(k)}$.

models for both functions and learning these through transitions experienced by the agent. These models can then be used to estimate the optimal Q with equation 3.1. However, this pipelined approach would result in errors propagating through the different stages of predictions.

A value iteration network (VIN) [108] abstracts away explicit computation of T and R by directly predicting the outcome of value iteration (Figure 3-4), thereby avoiding the aforementioned error propagation. The VI computation is mimicked by a recurrent network with two key operations at each step. First, to compute Q, we have two functions – f_R and f_T. f_R is a reward predictor that operates on $\phi(s, Z)$ while f_T utilizes the output of f_R and any previous V to predict Q. Both functions are parametrized as convolutional neural networks (CNNs),[2] to suit our tensor representation ϕ. Second, the network employs max pooling over the action channels in the Q-value map produced by f_T to obtain V. The value iteration computation

[2]Other parameterizations are possible for different input types, as noted in Tamar et al. [108].

(Eq. 3.1) can thus be approximated as:

$$\hat{Q}^{(n+1)}(s,a,Z) = f_T\Big(f_R(\phi(s,Z),a;\theta_R),\hat{V}^{(n)}(s,Z);\theta_T\Big) \tag{3.2}$$

$$\hat{V}^{(n+1)}(s,Z) = \max_a \hat{Q}^{(n+1)}(s,a,Z) \tag{3.3}$$

Note that while the VIN operates on $\phi(s,Z)$, we write $\hat{Q}$ and $\hat{V}$ in terms of the original state input s and text Z, since these are independent of our chosen representation.

The outputs of both CNNs are real-valued tensors – that of f_R has the same dimensions as the input state $(m \times n)$, while f_T produces $\hat{Q}^{(n+1)}$ as a tensor of dimension $m \times n \times |A|$. A key point to note here is that the model produces Q and V values for each cell of the input state matrix, assuming the agent's position to be that particular cell. The convolution filters help capture information from neighboring cells in our state matrix, which act as approximations for $V^{(n)}(s',Z)$. The parameters of the CNNs, θ_R and θ_T, approximate R and T, respectively. See Tamar et al. [108] for a more detailed discussion.

The recursive computation of traditional value iteration (Eq. 3.1) is captured by employing the CNNs in a recurrent fashion for k steps.[3] Intuitively, larger values of k imply a larger field of neighbors influencing the Q-value prediction for a particular cell as the information propagates longer. Note that the output of this recurrent computation, $\hat{Q}^{(k)}$, will be a 3-D tensor. However, since we need a policy only for the agent's current location, we use an appropriate selection function δ_s, which reduces this Q-value map to a single set of action values for the agent's location:

$$Q_{vin}(s,a,Z;\Theta_1) = \delta_s(\hat{Q}^{(k)}(s,a,Z)) \tag{3.4}$$

3.4.3 Final prediction

Games follow a complex dynamics which is challenging to capture precisely, especially longer term. VINs approximate the dynamics implicitly via learned convolutional operations. It is thus likely that the estimated Q_{vin} values are most helpful for short-term planning that corresponds to a limited number of iterations k. Therefore, we

[3] k is a model hyperparameter.

need to complement these ‘local’ Q-values with estimates based on a more global view.

To this end, following the VIN specification in [108], our architecture also contains a model-free action-value function, implemented as a deep Q-network (DQN) [73]. This network also provides a Q-value prediction – $Q_r(s, a, Z; \Theta_2)$ – which is combined with Q_{vin} using a composition function g[4]:

$$Q(s, a, Z; \Theta) = g(Q_{vin}(s, a, Z; \Theta_1), Q_r(s, a, Z; \Theta_2)) \tag{3.5}$$

The fusion of our model components enables our agent to establish the connection between input text descriptions, represented as vectors, and the environment's transitions and rewards, encoded as VIN parameters. In a new domain, the model can produce a reasonable policy using corresponding text, even before receiving any interactive feedback.

3.4.4 Parameter learning

Our entire model is end-to-end differentiable.We perform updates derived from the Bellman equation [105]:

$$Q_{i+1}(s, a, Z) \ = \mathrm{E}[r + \gamma \max_{a'} Q_i(s', a', Z) \mid s, a] \tag{3.6}$$

where the expectation is over all transitions from state s with action a and i is the update number. To learn our parametrized Q-function (the result of Eq. 3.5), we can use backpropagation through the network to minimize the following loss:

$$\mathcal{L}(\Theta_i) \ = \mathrm{E}_{\hat{s},\hat{a}}[(y_i - Q(\hat{s}, \hat{a}, Z; \Theta_i))^2] \tag{3.7}$$

where $y_i = r + \gamma \max_{a'} Q(s', a', Z; \Theta_{i-1})$ is the target Q-value with parameters Θ_{i-1} fixed from the previous iteration. We employ an experience replay memory $\mathcal{D}$ to store transitions [73], and periodically perform updates with random samples from this memory. We use an ϵ-greedy policy [105] for exploration.

[4]Although g can also be learned, we use component-wise addition in our experiments.

Algorithm 3.1 MULTITASK_TRAIN ($\mathcal{E}$)

```
1: Initialize parameters $\Theta$ and experience replay $\mathcal{D}$
2: for $k = 1, M$ do                                                    ▷ New episode
3:     Choose next environment $E_k \in \mathcal{E}$
4:     Initialize $E_k$; get start state $s_1 \in S_k$
5:     for $t = 1, N$ do                                                ▷ New step
6:         Select $a_t \sim$ EPS-GREEDY$(s_t, Q_\Theta, Z_k, \epsilon)$
7:         Execute action $a_t$, observe reward $r_t$ and new state $s_{t+1}$
8:         $\mathcal{D} = \mathcal{D} \cup (s_t, a_t, r_t, s_{t+1}, Z_k)$
9:         Sample mini batch $(s_j, a_j, r_j, s_{j+1}, Z_k) \sim \mathcal{D}$
10:        Perform gradient descent on loss $\mathcal{L}$ to update $\Theta$
11:        if $s_{t+1}$ is terminal then break
12: Return $\Theta$
```

Algorithm 3.2 EPS-GREEDY (s, Q, Z, ϵ)

```
1: if $random() < \epsilon$ then
2:     Return random action $a$
3: else
4:     Return $\arg\max_a \; Q(s, a, Z)$
5: Return $\Theta$
```

3.4.5 Transfer procedure

The traditional transfer learning scenario considers a single task in both source and target environments. To better test generalization and robustness of our methods, we consider transfer from *multiple* source tasks to *multiple* target tasks. We first train a model to achieve optimal performance on the set of source tasks. All model parameters (Θ) are shared between the tasks. The agent receives one episode at a time from each environment in a round-robin fashion, along with the corresponding text descriptions. Algorithm 3.1 details this multi-task training procedure.

After training converges, we use the learned parameters to initialize a model for the target tasks. All parameters of the VIN are replicated, while most weights of the representation generator are reused. Specifically, previously seen objects and words retain their learned entity-specific embeddings (v_o), whereas vectors for new objects/words in the target tasks are initialized randomly. All parameters are then fine-tuned on the target tasks, again with episodes sampled in a round-robin fashion.

3.5 Experimental Setup

3.5.1 Environments

We perform experiments on a series of 2-D environments within the GVGAI framework [88], which is used for an annual video game AI competition.[5] In addition to pre-specified games, the framework supports the creation of new games using the Py-VGDL description language [96]. We use four different games to evaluate transfer and multitask learning: *Freeway*, *Bomberman*, *Boulderchase* and *Friends & Enemies*. There are certain similarities between these games. For one, each game consists of a 16x16 grid with the player controlling a movable avatar with two degrees of freedom. Also, each domain contains other entities, both stationary and moving (e.g. diamonds, spiders), that can interact with the avatar.

However, each game also has its own distinct characteristics. In *Freeway*, the goal is to cross a multi-lane freeway while avoiding cars in the lanes. The cars move at various paces in either horizontal direction. *Bomberman* and *Boulderchase* involve the player seeking an exit door while avoiding enemies that either chase the player, run away or move at random. The agent also has to collect resources like diamonds and dig or place bombs to clear paths. These three games have five level variants each with different map layouts and starting entity placements.

Friends & Enemies (F&E) is a new environment we designed, with a larger variation of entity types. This game has a total of twenty different non-player entities, each with different types of movement and interaction with the player's avatar. For instance, some entities move at random while some chase the avatar or shoot bullets that the avatar must avoid. The objective of the player is to meet all friendly entities while avoiding enemies. For each game instance, four non-player entities are sampled from this pool and randomly placed in the grid. This makes F&E instances significantly more varied than the previous three games. We created two versions of this game: F&E-1 and F&E-2, with the sprites in F&E-2 moving faster, making it a

[5] http://www.gvgai.net/

harder environment. Table 3.1 shows all the different transfer scenarios we consider in our experiments.

Condition	Source	Target
F&E-1 $\rightarrow$ F&E-2	7	3
F&E-1 $\rightarrow$ Freeway	7	5
Bomberman $\rightarrow$ Boulderchase	5	5

Table 3.1: Number of source and target game instances for different transfer conditions.

3.5.2 Text descriptions

We collect textual descriptions using Amazon Mechanical Turk [18]. We provide annotators with sample gameplay videos of each game and ask them to describe specific entities in terms of their movement and interactions with the avatar. Since we ask the users to provide an independent account of each entity, we obtain "descriptive" sentences as opposed to "instructive" ones which inform the optimal course of action from the avatar's viewpoint. From manual verification, we find less than 3% of the obtained annotations to be "instructive".

We aggregated together four sets of descriptions, each from a different annotator, for every environment. Each description in an environment is aligned to one constituent entity. We also make sure that the entity names are not repeated across games (even for the same entity type). Table 3.2 provides corpus-level statistics on the collected data and Figure 3-2 has sample descriptions.

Unique word types	286
Avg. words / sentence	8.65
Avg. sentences / domain	36.25
Max sentence length	22

Table 3.2: Overall statistics of the text descriptions collected using Mechanical Turk.

3.5.3 Evaluation metrics

We evaluate transfer performance using three metrics employed in previous approaches [110]:

- *Average Reward*, which is area under the reward curve divided by the number of test episodes.
- *Jumpstart performance*, which is the average reward over first 100k steps.
- *Asymptotic performance*, which is the average reward over 100k steps after convergence.

For the multitask scenario, we consider the average and asymptotic reward only. We repeat all experiments with three different random seeds and report average numbers.

3.5.4 Baselines

We explore several baselines:

- NO TRANSFER: A deep Q-network (DQN) [73] is initialized at random and trained from scratch on target tasks. This is the only case that does not use parameters transferred from source tasks.
- DQN: A DQN is trained on source tasks and its parameters are transferred to target tasks. This model does not make use of text descriptions.
- TEXT-DQN: DQN with hybrid representation $\phi(s, Z)$, using text descriptions. This is essentially a reactive-only version of our model, i.e. without the VIN planning module.
- AMN: Actor-Mimic network, a recently proposed [85] transfer method for deep RL using policy distillation.[6]

3.5.5 Implementation details

For all models, we set $\gamma = 0.8$, $\mathcal{D} = 250$k. We used the *Adam* [55] optimization scheme with a learning rate of 10^{-4}, annealed linearly to 5×10^{-5}. The minibatch size was set to 32. ϵ was annealed from 1 to 0.1 in the source tasks and set to 0.1

[6]We only evaluate AMN on transfer since it does not perform online multitask learning and is not directly comparable.

in the target tasks. For the value iteration module (VIN), we experimented with different levels of recurrence, $k \in \{1, 2, 3, 5\}$ and found $k = 1$ or $k = 3$ to work best.[7] For DQN, we used two convolutional layers followed by a single fully connected layer, with ReLU non-linearities.The CNNs in the VIN had filters and strides of length 3. The CNNs in the model-free component used filters of sizes $\{4, 2\}$ and corresponding strides of size $\{3, 2\}$. All embeddings are initialized to random values.[8]

[7]We still observed transfer gains with all k values.

[8]We also experimented with using pre-trained word embeddings for text but obtained equal or worse performance.

3.6 Results

3.6.1 Transfer performance

Table 3.3 demonstrates that transferring policies positively assists learning in new domains. Our model, TEXT-VIN, performs at par or better than the baselines across all the different metrics. On the first metric of *average reward*, TEXT-VIN (1) achieves a 8% gain (absolute) over AMN on F&E-1 $\rightarrow$ F&E-2, while TEXT-VIN (3) achieves a 35% gain (absolute) over TEXT-DQN on F&E-1 $\rightarrow$ Freeway. This is also evident from a sample reward curve, shown in Figure 3-5 (left).

In *jumpstart* evaluation, all the transfer approaches outperform the NO TRANSFER baseline, except for AMN on Bomberman $\rightarrow$ Boulderchase. In all domains, the TEXT-DQN model obtains higher scores than DQN, already demonstrating the advantage of using text for transfer. TEXT-VIN (3) achieves the highest numbers in all transfer settings, demonstrating effective utilization of text descriptions to bootstrap in a new environment.

On the final metric of *asymptotic performance*, TEXT-VIN achieves the highest convergence scores, except on F&E-1 $\rightarrow$ F&E-2, where AMN obtains a score of 1.64. This is in part due to its smoother convergence[9]; improving the stability of our model training could lead to improved asymptotic performance.

Negative transfer We also observe the challenging nature of policy transfer in some scenarios. For example, in Bomberman $\rightarrow$ Boulderchase, TEXT-DQN and AMN achieve a *lower* average reward and *lower* asymptotic reward than the NO TRANSFER model, exhibiting negative transfer [110]. Further, TEXT-DQN performs worse than a vanilla DQN in such cases, which further underlines the need for a model-aware approach to truly take advantage of the descriptive text.

[9]This fact is also noted in [85]

Model	F&E-1 → F&E-2			F&E-1 → Freeway			Bomberman → Boulderchase		
	Avg.	*Jumpstart*	*Asymp.*	*Avg.*	*Jumpstart*	*Asymp.*	*Average*	*Jumpstart*	*Asymp.*
NO TRANSFER	0.86	-0.19	1.40	0.15	-1.06	0.81	9.50	2.88	10.99
DQN	1.02	0.73	1.30	0.06	-0.96	0.82	9.63	3.84	11.28
TEXT-DQN	1.03	0.40	1.33	0.38	-0.50	**0.85**	8.52	3.42	9.45
AMN (Actor Mimic)	1.22	0.13	**1.64**	0.08	-0.84	0.75	6.22	0.78	8.20
TEXT-VIN (1)	**1.38**	0.93	1.53	0.63	-0.58	**0.85**	**11.41**	4.42	12.06
TEXT-VIN (3)	1.27	**1.04**	1.44	**0.73**	**-0.01**	**0.85**	10.93	**4.49**	**12.09**

Table 3.3: Transfer learning results under the various metrics for different domains (**Avg.** is average reward over time, **Asymp.** is asymptotic reward). Numbers in parentheses for TEXT-VIN indicate k value. TEXT- models make use of textual descriptions. The max reward attainable (ignoring step penalties) in the target environments is 2.0, 1.0 and at least 25.0 in F&E, Freeway and Boulderchase, respectively. Higher scores are better; bold indicates best numbers.

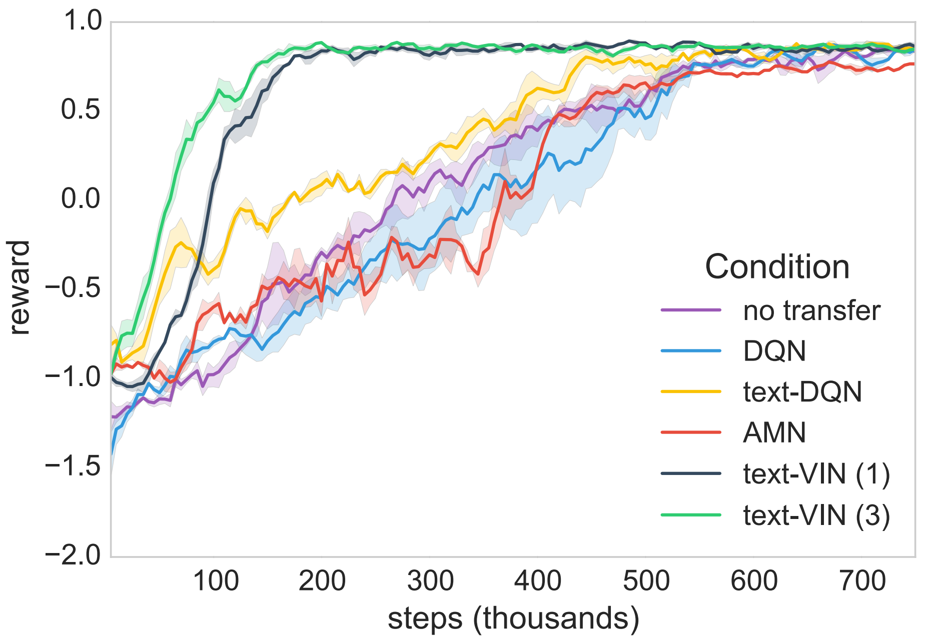


Figure 3-5: Reward curve for transfer condition F&E-1 $\rightarrow$ Freeway. Numbers in parentheses for TEXT-VIN indicate k value. All graphs were averaged over 3 runs with different seeds, with shaded areas representing bootstrapped confidence intervals. Curves for other transfer conditions are provided in Appendix B.

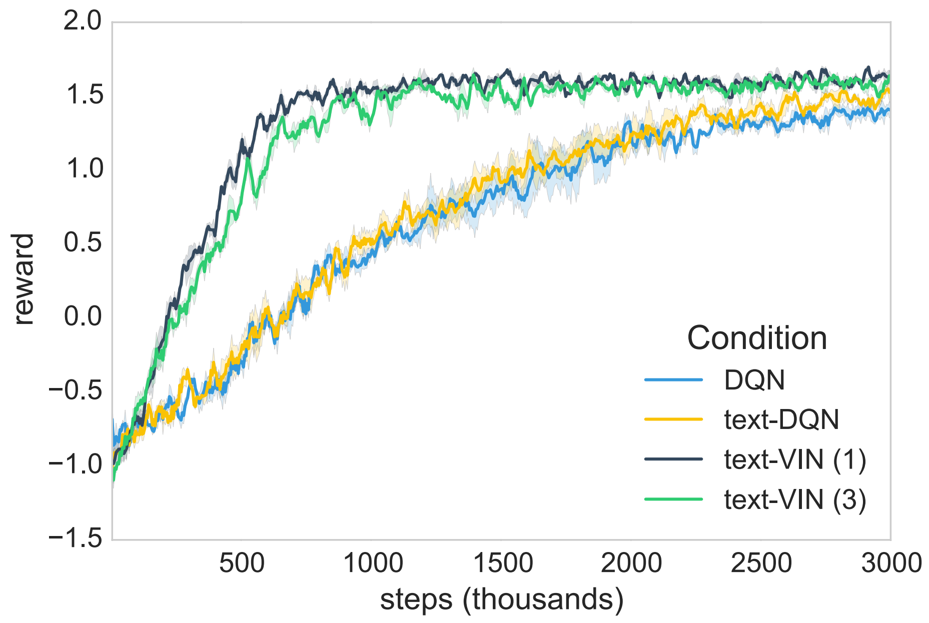


Figure 3-6: Reward curve for multitask learning in F&E-2. Numbers in parentheses for TEXT-VIN indicate k value. All graphs were produced by averaging over 3 runs with different seeds; shaded areas represent bootstrapped confidence intervals.

Model	*Avg*	*Asymp.*
DQN	0.65	1.38
TEXT-DQN	0.71	1.49
TEXT-VIN (1)	**1.32**	**1.63**
TEXT-VIN (3)	1.24	1.57

Table 3.4: Scores for multitask learning over 20 games in F&E-2.

3.6.2 Multi-task performance

The learning benefits observed in the transfer scenario are also present in the multi-task setup. Table 3.4 details the average reward and asymptotic reward for learning across twenty variants of the F&E-2 domain, simultaneously. Our model utilizes the text to learn faster and achieve higher optimum scores, with TEXT-VIN (1) showing gains over TEXT-DQN of 30% and 7% on average and asymptotic rewards, respectively. Figure 3-5 (right) shows the corresponding reward curves.

3.6.3 Analysis

Effect of factorized representation We investigate the usefulness of our factorized representation by training a variant of our model using only a text-based vector representation (*Text only*) for each entity. We consider two different transfer scenarios – (a) when both source and target instances are from the same domain (F&E-1 $\rightarrow$ F&E-1) and (b) when the source/target instances are in different domains (F&E-1 $\rightarrow$ F&E-2). In both cases, we see that our two-part representation results in faster learning and more effective transfer, obtaining 20% higher average reward and 16% more asymptotic reward in F&E-1 $\rightarrow$ F&E-2 transfer. Our representation is able to transfer prior knowledge through the text-based component while retaining the ability to learn new entity-specific representations quickly.

Text representation: Sum vs LSTM We also consider a different variant of our model that uses a simple sum of the word embeddings for a description, instead of an LSTM. Table 3.6 provides a comparison of transfer performance on one of the

Condition	Model	*Avg*	*Jump*	*Asymp.*
F&E-1 → F&E-1	Text only	1.64	0.48	**1.78**
	Text+entity ID	**1.70**	**1.09**	**1.78**
F&E-1 → F&E-2	Text only	1.05	0.62	1.15
	Text+entity ID	**1.27**	**1.04**	**1.44**

Table 3.5: Transfer results using different input representations with TEXT-VIN (3). *Text only* means only a text-based vector is used, i.e. $\phi(s) = v_z(s, Z)$. *Text+entity ID* refers to our full representation, $\phi(s) = [v_z(s, Z); v_o(s)]$.

Model	Sum	LSTM
TEXT-DQN	6.57	**8.52**
TEXT-VIN (1)	9.26	**11.41**
TEXT-VIN (2)	9.17	**10.78**
TEXT-VIN (3)	10.63	**10.93**
TEXT-VIN (5)	9.54	**10.15**

Table 3.6: Average rewards in Bomberman → Boulderchase with different text representations: **Sum** of word vectors, or an **LSTM**-based recurrent neural network over the entire sentence.

conditions. We observe that across all models, the LSTM representation provides greater gains. In fact, the Sum representation does worse than vanilla DQN (9.63) in some cases. This underscores the importance of a good text representation for the model.

Value analysis Finally, we provide some qualitative evidence to demonstrate the generalization capacity of TEXT-VIN. Figure 3-7 shows visualizations of four value maps produced by the VIN module of a trained model, with the agent's avatar at position (4,4) and a single entity at (2,6) in each map. In the first map, the entity is known and friendly, which leads to high values in the surrounding areas, as expected. In the second map, the entity is unseen and without any descriptions; hence, the values are uninformed. The third and fourth maps, however, contain unseen entities with descriptions. In these cases, the module predicts higher or lower values around the entity depending on whether the text portrays it as a friend or enemy. Thus, even before a single interaction in a new domain, our model can utilize text to generate good value maps. This bootstraps the learning process, making it more efficient.

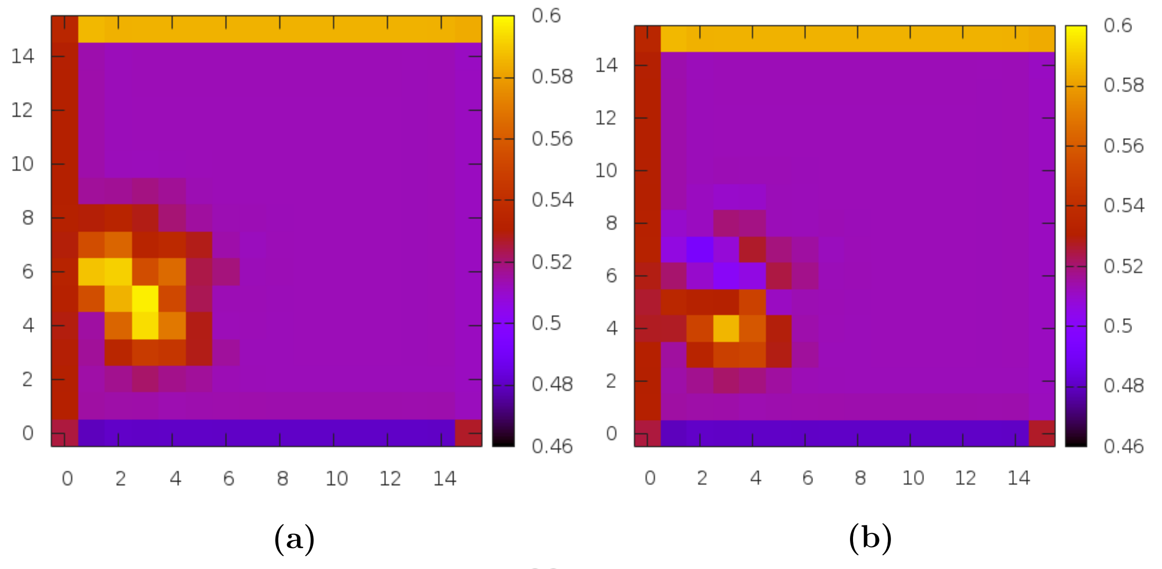
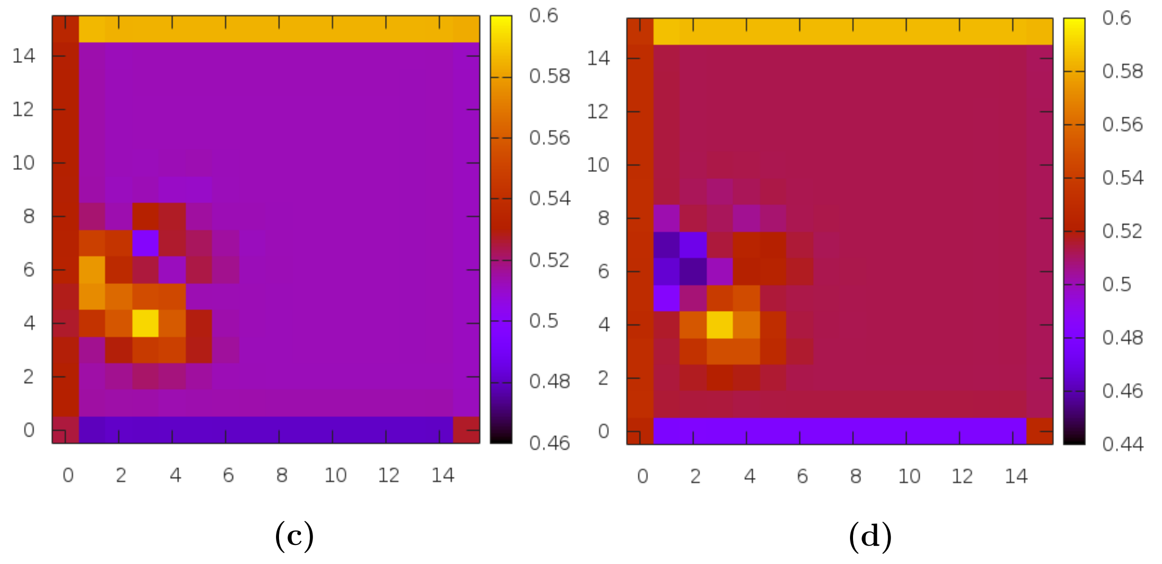



Figure 3-7: Value maps, $\hat{V}^{(k)}(s, Z)$, produced by the VIN module for (a) seen entity (friend), (b) unseen entity with no description, (c) unseen entity with 'friendly' description, and (d) unseen entity with 'enemy' description. Agent is at (4,4) and the non-player entity is at (2,6). Note how the model can construct a reasonably accurate value map for unseen entities using their descriptions.

3.7 Conclusions

We have proposed a novel method of utilizing natural language to drive transfer for reinforcement learning (RL). We show that textual descriptions of environments provide a compact intermediate channel to facilitate effective policy transfer. Using a model-aware RL approach, our design consists of a differentiable planning module (VIN), a model-free component and a two-part representation to effectively utilize entity descriptions. We demonstrate the effectiveness of our approach on both transfer and multi-task scenarios in a variety of environments. We discuss a few research directions that can be pursued based on this work.

Future work

- Using language for policy transfer in deep RL is complementary to other techniques such as policy reuse [39] or skill transfer [42] among other approaches [31, 112, 123]. A combination of one of these methods with language-guided transfer could result in further improvements.

- A major factor in successfully utilizing the text descriptions for transfer was the model-based VIN component. However, in the current form, this requires specifying the recurrence hyper-parameter k, whose optimal value might vary from one domain to another. An interesting research direction is to investigate models that can perform multiple levels of recurrent VI computation, possibly in a dynamic fashion. This would allow an agent to plan and act over multiple temporal scales.

Chapter 4

Improved Information Extraction with Reinforcement Learning

In this chapter, we demonstrate the utility of sequential decision making to enhance systems for information extraction, a traditional NLP task. We design a system that can automatically issue queries on the web, retrieve external articles and aggregate information from multiple sources. We train the system using Reinforcement Learning to maximize the accuracy of the final extractions. Our experiments demonstrate a clear advantage in exploiting redundant sources of information, resulting in more robust extractions.

4.1 Introduction

The problem of information extraction (IE) has remained a mainstay of traditional natural language processing over the last few decades [70, 21, 95, 1]. At its core, the task involves wresting out structured pieces of knowledge from raw, unstructured text documents. Many approaches to the problem exist, ranging from manually created rules [75] and fully supervised extraction models [94, 86] to semi-supervised approaches [25, 53, 19]. In the latter two statistical approaches, the key idea is in learning to identify patterns of text that help extract the values of interest. The more

A couple and four children found dead in their burning South Dakota home had been shot in an apparent murder-suicide, officials said Monday.
...
The State Attorney General's office said Monday that preliminary autopsy results identified cause of death for Nicole Westerhuis and her children Nicole, Kailey, Jaeci, Connor and Michael as "homicide by shotgun." **Scott Westerhuis's** cause of death was "shotgun wound with manner of death as suspected suicide," it added in a statement.

ShooterName: Scott Westerhuis
NumKilled: 6
NumWounded: 0
City: Platte

Figure 4-1: Sample news article (**top**) and desired extractions (**bottom**) for a mass shooting case in the United States. The article contains both the name of the shooter as well as the number of people killed, but both pieces of information require complex extraction techniques (e.g. reasoning 'a couple and four children' equals six, or 'suspected suicide' implies the shooter was Scott Westerhuis).

patterns an IE system can identify, the more accurate it is likely to be.

In many realistic domains, information extraction (IE) systems require exceedingly large amounts of annotated data to deliver high performance. Increases in the amount of training data enable models to robustly handle the multitude of linguistic expressions that convey the same semantic relation. Consider, for instance, an IE system that aims to identify entities such as the perpetrator and the number of victims in a shooting incident (Figure 4-1). The document does not explicitly mention the shooter (*Scott Westerhuis*), but instead refers to him as a suicide victim. Extraction of the number of fatally shot victims is similarly difficult, as the system needs to infer that *"A couple and four children"* means *six people*. Even a large annotated training set may not provide sufficient coverage to capture such challenging cases.

In this chapter, we explore an alternative approach for boosting extraction accuracy, when a large training corpus is not available. The proposed method utilizes external information sources to resolve ambiguities inherent in text interpretation.

> The **six** members of a South Dakota family found dead in the ruins of their burned home were fatally shot, with one death believed to be a suicide, authorities said Monday.

> AG Jackley says all evidence supports the story he told based on preliminary findings back in September: **Scott Westerhuis** shot his wife and children with a shotgun, lit his house on fire with an accelerant, then shot himself with his shotgun.

Figure 4-2: Two other articles on the same shooting case shown in Figure 4-1. The first article clearly mentions that six people were killed. The second one portrays the shooter in an easily extractable form.

Specifically, our strategy is to find other documents that contain the information sought, expressed in a form that a basic extractor can "understand". For instance, Figure 4-2 shows two other articles describing the same event, wherein the entities of interest – the number of people killed and the name of the shooter – are expressed explicitly. Processing such stereotypical phrasing is easier for most extraction systems, compared to analyzing the original source document. This approach is particularly suitable for extracting information from news articles, since a typical event is often covered by multiple news outlets.

There are two main challenges to this approach. First, we have the task of performing *event coreference*, which entails retrieving suitable articles describing the same incident as our original document. Querying the web (using the source article's title for instance) often retrieves documents about other incidents with a tangential relation to the original story. For example, the query "*4 adults, 1 teenager shot in west Baltimore 3 april 2015*" yields only two relevant articles among the top twenty results on Bing search, while returning other shooting events at the same location. The second challenge is that of *value reconciliation.* Even when we have two articles on the same event, our extraction system can provide us with conflicting values for each slot. It is very likely that some of these, if not all, are inaccurate. Therefore, we require a good mechanism for resolving the differing predictions and producing a single final answer.

One solution to this problem would be to perform a single web search to retrieve articles on the same event and then reconcile values extracted from them (say, using another classifier). However, if the confidence of the new set of values is still low, we might wish to perform further queries. Further, the queries we utilize may vary depending on the type of information we seek to obtain (e.g. the values the system is currently least confident about). Thus, the problem is inherently sequential, requiring alternating phases of querying to retrieve articles and integrating the extracted values.

We address these challenges using a Reinforcement Learning (RL) approach that combines query selection, extraction from new sources, and value reconciliation. To effectively select among possible actions, our state representation encodes information about the current and new entity values along with the similarity between the source article and the newly retrieved document. The model learns to select good actions for both querying and value reconciliation in order to optimize the reward function, which reflects extraction accuracy and includes penalties for extra moves. We train the RL agent using a Deep Q-Network (DQN) [73] as a value function approximator. The DQN is used to predict both querying and reconciliation choices simultaneously. While we use a maximum entropy model as the base extractor, this framework can be inherently used with other extraction algorithms.

We evaluate our system on two datasets where available training data is inherently limited. The first dataset is constructed from a publicly available database of mass shootings in the United States. The database is populated by volunteers and includes the source articles. The second dataset is derived from a FoodShield database of illegal food adulterations. Our experiments demonstrate that the final RL model outperforms basic extractors as well as a meta-classifier baseline in both domains. For instance, in the *Shootings* domain, the average accuracy improvement over the meta-classifier is 7%.

The rest of this chapter is structured as follows. Section 4.2 discusses prior related work on information extraction, entity linking and knowledge base completion. In Section 4.3, we describe our entire MDP framework for learning to query and aggregate external evidence. Section 4.4 details our model and learning procedure. Section 4.5 explains our experimental setup, and Section 4.6 presents the results along with some analysis. Finally, we conclude in Section 4.7 and provide some suggestions for future work.

4.2 Related Work

4.2.1 Open Information Extraction

Existing work in open IE has used external sources from the web to improve extraction accuracy and coverage [2, 34, 35, 120]. Such research has focused on identifying multiple instances of the same relation, independent of the context in which this information appears. In contrast, our goal is to extract information from additional sources about a specific event described in a source article. Therefore, the novel challenge of our task resides in performing event coreference [63, 10] (i.e identifying other sources describing the same event) while simultaneously reconciling extracted information. Moreover, relations typically considered by open IE systems have significantly higher coverage in online documents than a specific incident described in a few news sources. Hence, we require a different mechanism for finding and reconciling online information.

4.2.2 Entity linking, multi-document extraction and event coreference

Our work also relates to the task of multi-document information extraction, where the goal is to connect different mentions of the same entity across input documents [68, 20, 44, 32]. Since this setup already includes multiple documents, an extraction system is not required to look for additional sources or decide on their relevance. Also, while the set of input documents overlap in terms of entities mentioned, they do not necessarily describe the same event, as is the case in our setup. Given these differences, the challenges and opportunities of our framework are distinct from these works.

4.2.3 Knowledge Base Completion and Online Search

Recent work has explored several techniques to perform Knowledge Base Completion (KBC) such as vector space models and graph traversal [102, 122, 36, 80, 43].

Though our work also aims at increasing extraction recall for a database, traditional KBC approaches do not require searching for additional sources of information. Work by West et al. [118] explores query reformulation in the context of KBC. Using existing search logs, they learn how to formulate effective queries for different types of database entries. Once query learning is completed, the model employs several selected queries, and then aggregates the results based on retrieval ranking. This approach is complementary to the proposed method, and can be combined with our approach if search logs are available.

Kanani and McCallum [54] also combine search and information extraction. In their task of faculty directory completion, the system has to find documents from which to extract desired information. They employ reinforcement learning to address computational bottlenecks, by minimizing the number of queries, document downloads and extraction action. The extraction accuracy is not part of this optimization, since the baseline IE system achieves high performance on the relations of interest. Hence, given different design goals, the two RL formulations are very different. Our approach is also close in spirit to the AskMSR system [9] which aims at using information redundancy on the web to answer questions better. Several slot-filling methods have also experimented with query formulation over web-based corpora to populate knowledge bases [103, 52]. Though our goal is similar, we learn to query and consolidate the different sources of information instead of using pre-defined rules.

4.3 Framework

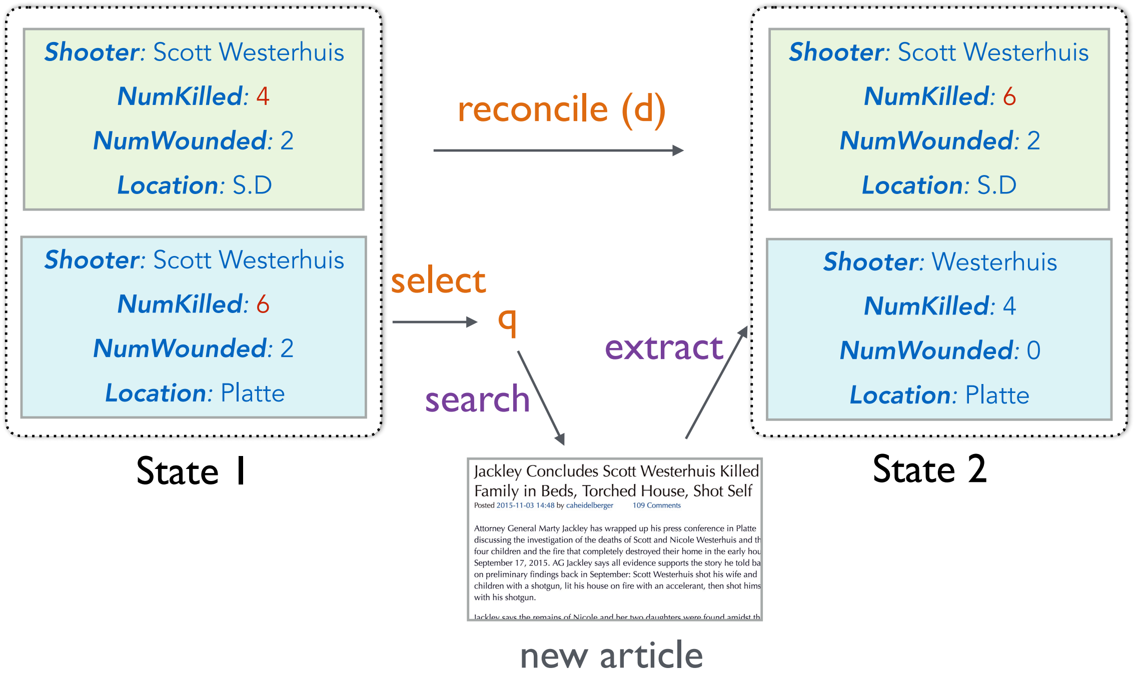
Our main goal is to augment traditional IE systems with the capability to autonomously locate additional sources on the web, determine their relevance, and aggregate information reliably. The kind of external document sought, however, depends on the particular type of information the system is looking for. This could be a fact that is missing in the original article, or a value that the system has low confidence on. For instance, if the name of the shooter is missing, the system should probably look for articles containing words such as *shooter, perpetrator, gunman*, etc. Further, we would like the system to retrieve and process a small number of extra documents to minimize the cost associated. These requirements point towards a framework that enables sequential decision making.

To this end, we model the information extraction task as a markov decision process (MDP), where the model learns to utilize external sources to improve upon extractions from a source article (see Figure 4-3). The MDP framework allows us to dynamically incorporate entity predictions while also providing flexibility to choose the type of articles to extract from. At each step, the system has to reconcile the extracted values from a new article with the current set of values, and then decide on the next query for retrieving more articles. We represent the MDP as a tuple $\langle S, A, T, R \rangle$, where S is the space of all possible states, $A = \{a = (d, q)\}$ is the set of all actions, $R(s, a)$ is the reward function, and $T(s'|s, a)$ is the transition function. We describe these in detail below.

4.3.1 States

A state $s \in S$ in our MDP consists of the extractor's confidence in predicted entity values, the context from which the values are extracted and the similarity between the new document and the original one. We represent the state as a continuous real-valued vector (Figure 4-4) incorporating these pieces of information:

1. Confidence scores of current and newly extracted entity values.



Jackley Concludes Scott Westerhuis Killed Family in Beds, Torched House, Shot Self
Posted 2015-11-03 14:48 by caheidelberger 109 Comments

Attorney General Marty Jackley has wrapped up his press conference in Platte discussing the investigation of the deaths of Scott and Nicole Westerhuis and th four children and the fire that completely destroyed their home in the early hou September 17, 2015. AG Jackley says all evidence supports the story he told ba on preliminary findings back in September: Scott Westerhuis shot his wife and children with a shotgun, lit his house on fire with an accelerant, then shot hims with his shotgun.

Jackley says the remains of Nicole and her two daughters were found amidst th

Figure 4-3: Illustration of a transition in the MDP – the top box in each state shows the current entities and the bottom one consists of the new entities extracted from a downloaded article on the same event. At each step, the system makes two choices – a reconciliation decision (d) and a query selection (q), resulting in a corresponding evolution of the state.

2. One-hot encoding of matches between current and new values.

3. *tf-idf* similarity between the original article and the new article.

4. Unigram/tf-idf counts[1] of context words. These are words that occur in the neighborhood of the entity values in a document (e.g. the words *which*, *left*, *people* and *wounded* in the phrase "*which left 5 people wounded*").

These four parts serve as signals for our system to use in performing both event coreference (#3,#4) and value reconciliation (#1,#2,#4).

4.3.2 Actions

We design our system to jointly reconcile values and perform event coreference. At each step, the agent is required to take two actions - a reconciliation decision d,

[1]Counts are computed on the documents used to train the basic extraction system.

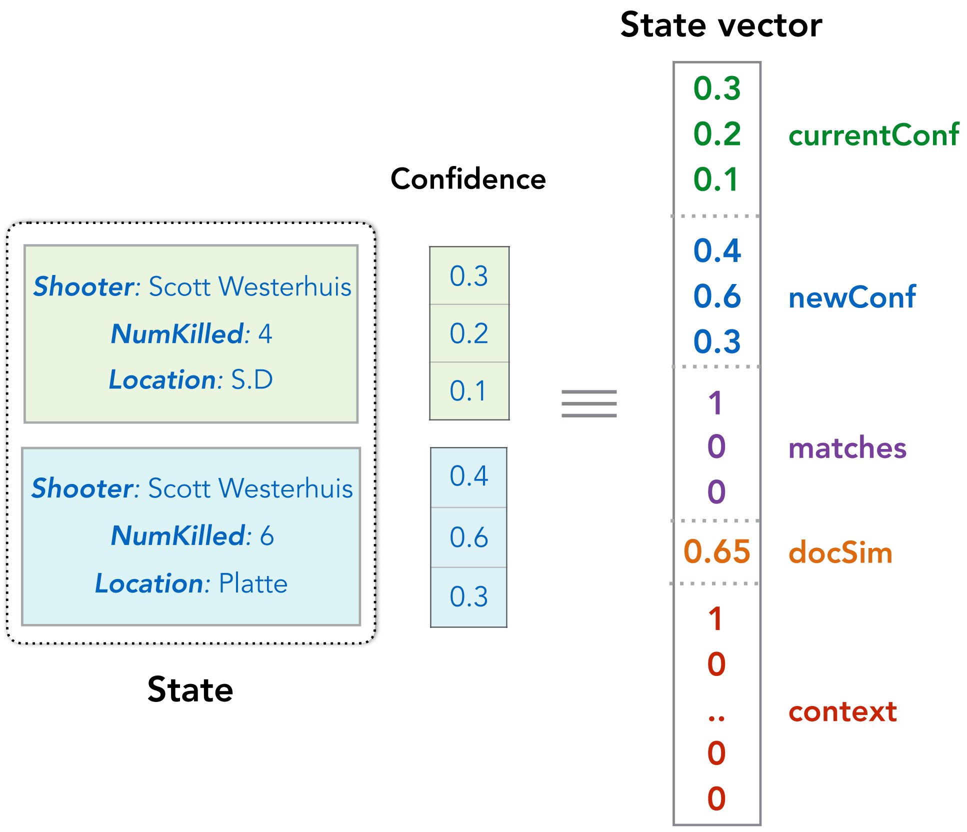


Figure 4-4: Sample state representation (**right**) in the MDP based on current and new values of entities (**left**). The different parts of the state are – *currentConf*: confidence scores of current entities, *newConf*: confidence scores of new entities, *matches* between current and new values, *docSim*: tf-idf similarity between original and currently inspected document, and *context*: tf-idf counts of context words.

and a query choice q. The decision d on the newly extracted values can be one of the following types: (1) accept a specific entity's value (one action per entity)[2], (2) accept all entity values, (3) reject all values or (4) stop. In cases 1-3, the agent continues to inspect more articles, while the episode ends if a stop action (4) is chosen. The current values and confidence scores are simply updated with the accepted values and the corresponding confidences.[3] We restrict ourselves to these choices instead of all possible subsets due to computational constraints. The choice q is used to select the next query from a set of automatically generated alternatives (details in Section 4.3.4) in order to retrieve the next article.

4.3.3 Rewards

The reward function is chosen to maximize the final *extraction accuracy* while minimizing the number of queries. The accuracy component is calculated using the difference between the accuracy of the current and the previous set of entity values:

$$R(s,a) = \sum_{\text{entity } j} \text{Acc}(e^j_{cur}) - \text{Acc}(e^j_{prev}) + \delta$$

There is also a negative reward per step (δ) to penalize the agent more for longer episodes.

4.3.4 Queries

The queries used by our model are based on automatically generated templates, created using the title of an article along with words[4] most likely to co-occur with each entity type in the training data. Table 4.1 provides some examples – for instance, the second template contains words such as *arrested* and *identified* which often appear around the name of the shooter. Templates for the domain of food adulteration are provided in Appendix C.

[2]No entity-specific features are used for action selection.

[3]We also experiment with other forms of value reconciliation. See Section 4.5 for details.

[4]Stop words, numeric terms and proper nouns are filtered.

$\langle$*title*$\rangle$
$\langle$*title*$\rangle$ + (police | identified | arrested | charged)
$\langle$*title*$\rangle$ + (killed | shooting | injured | dead | people)
$\langle$*title*$\rangle$ + (injured | wounded | victim)
$\langle$*title*$\rangle$ + (city | county | area)

Table 4.1: Examples of different query templates for web search for articles on mass shootings. The last four queries contain context words around values for entity types ShooterName, NumKilled, NumWounded and City, respectively. The | symbol represents logical OR. At query time, $\langle$*title*$\rangle$ is replaced by the source article's title.

We employ a search engine to query the web for articles, using one of the filled-in query templates, and then retrieve the top k links per query.[5] Documents that are more than a month older than the original article are filtered out of the search results.

4.3.5 Transitions

Each episode starts off with a single source article x_i, from which an initial set of entity values (e_0) are extracted. The subsequent steps in the episode involve extra articles from the web, downloaded using different query formulations based on the source article. A single transition in the episode consists of the agent being given the state s containing information about the current and new set of values (extracted from a single article) using which the next action $a = (d, q)$ is chosen. The transition function $T(s'|s, a)$ is responsible for incorporating two effects – (1) the reconciliation decision d from the agent is used to combine the two sets of values in state s into the first set of values in the next state s', and (2) the query q is employed to retrieve the next article, from which values are extracted to produce the second set for s'. The episode stops whenever d is a stop decision.

Algorithm 4.1 details the working of the MDP framework during the training phase. During the test phase, each source article is handled only once in a single episode (lines 8-23).

[5]We use k=20 in our experiments.

Algorithm 4.1 MDP framework for Information Extraction (Training Phase)

```
1: Initialize set of original articles $X$
2: for $x_i \in X$ do
3:     for each query template $T^q$ do
4:         Download articles with query $T^q(x_i)$
5:         Queue retrieved articles in $Y_i^q$
6: for $epoch = 1, M$ do
7:     for $i = 1, |X|$ do                                    ▷ New event instance
8:         Extract entities $e_0$ from $x_i$
9:         $e_{cur} \leftarrow e_0$
10:        $q \leftarrow 0$, $r \leftarrow 0$                  ▷ Variables for query type, reward
11:        while $Y_i^q$ not empty do
12:            Pop next article $y$ from $Y_i^q$
13:            Extract entities $e_{new}$ from $y$
14:            Compute tf-idf similarity $\mathcal{Z}(x_i, y)$
15:            Compute context vector $\mathcal{C}(y)$
16:            Form state $s$ using $e_{cur}$, $e_{new}$, $\mathcal{Z}(x_i, y)$ and $\mathcal{C}(y)$
17:            Send $(s, r)$ to agent
18:            Get decision $d$, query $q$ from agent
19:            if $d == \text{STOP}$ then break
20:            $e_{prev} \leftarrow e_{cur}$
21:            $e_{cur} \leftarrow Reconcile(e_{cur}, e_{new}, d)$
22:            $r \leftarrow \sum_{\text{entity } j} \text{Acc}(e^j_{cur}) - \text{Acc}(e^j_{prev})$
23:        Send $(s_{end}, r)$ to agent
```

4.4 Reinforcement Learning for Information Extraction

4.4.1 Model

To learn a good policy for an agent, we utilize the paradigm of reinforcement learning (RL). The MDP described in the previous section can be viewed in terms of a sequence of transitions (s, a, r, s'). The agent typically estimates a state-action value function $Q(s, a)$ to determine which action a to perform in state s. A commonly used technique for learning an optimal value function is Q-learning [117], in which the agent iteratively updates $Q(s, a)$ using the rewards obtained from episodes. The updates are derived from the recursive Bellman equation [105] for the optimal Q:

$$Q_{i+1}(s, a) \ = \mathrm{E}[r + \gamma \max_{a'} Q_i(s', a') \mid s, a]$$

Here, $r = R(s, a)$ is the reward and γ is a factor discounting the value of future rewards. The expectation is over all transitions involving state s and action a.

Since our problem involves a continuous state space S, we use a deep Q-network (DQN) [73] as a function approximator, $Q(s, a) \approx Q(s, a; \theta)$. The DQN, in which the Q-function is approximated using a deep neural network, has been shown to learn better value functions than linear approximators [78, 46] and can capture non-linear interactions between the different pieces of information in our state.

We use a DQN consisting of two linear layers (20 hidden units each) followed by rectified linear units (ReLU), along with two separate output layers.[6] The network takes the continuous state vector s as input and predicts $Q(s, d)$ and $Q(s, q)$ for reconciliation decisions d and query choices q simultaneously using the different output layers (see Figure 4-5 for the model architecture).

[6]We did not observe significant differences with additional linear layers or the choice of non-linearity (Sigmoid/ReLU).

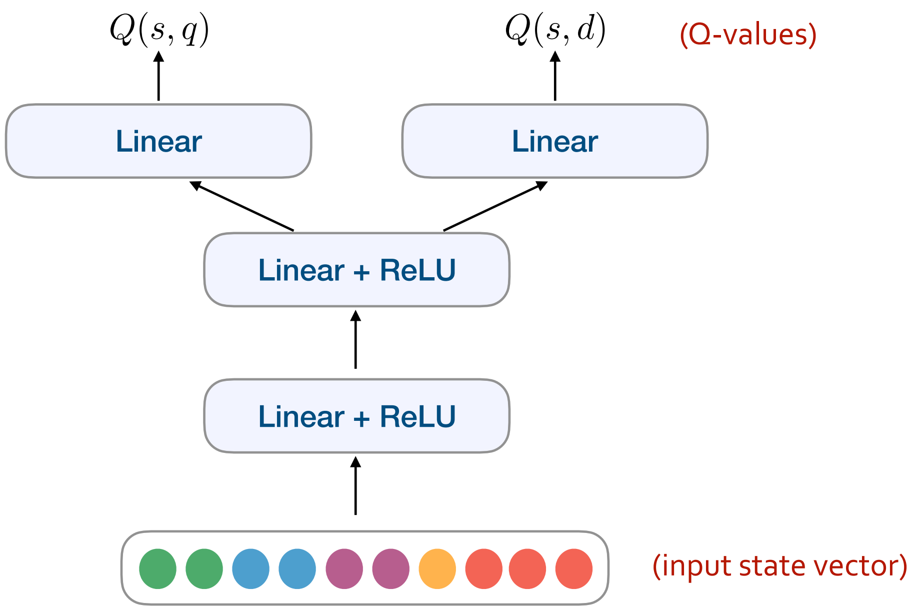


Figure 4-5: Architecture of the deep Q-network (DQN), whose input is the state vector described in Section 4.3.1 and outputs are Q-values for both query and reconciliation decisions.

Algorithm 4.2 Training Procedure for DQN agent with ϵ-greedy exploration

```
 1: Initialize experience memory $\mathcal{D}$
 2: Initialize parameters $\theta$ randomly
 3: for $episode = 1, M$ do
 4:     Initialize environment and get start state $s_1$
 5:     for $t = 1, N$ do
 6:         if $random() < \epsilon$ then
 7:             Select a random action $a_t$
 8:         else
 9:             Compute $Q(s_t, a)$ for all actions $a$
10:             Select $a_t = \text{argmax}\ Q(s_t, a)$
11:         Execute action $a_t$ and observe reward $r_t$ and new state $s_{t+1}$
12:         Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$
13:         Sample random mini batch of transitions $(s_j, a_j, r_j, s_{j+1})$ from $\mathcal{D}$
14:         $y_j = \begin{cases} r_j, & \text{if } s_{j+1} \text{ is terminal} \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta_t), & \text{else} \end{cases}$
15:         Perform gradient descent step on the loss $\mathcal{L}(\theta) = (y_j - Q(s_j, a_j; \theta))^2$
16:         if $s_{t+1} == s_{end}$ then break
```

4.4.2 Parameter Learning

The parameters θ of the DQN are learned using stochastic gradient descent with RMSprop [111]. Each parameter update aims to close the gap between the $Q(s_t, a_t; \theta)$ predicted by the DQN and the expected Q-value, $r_t + \gamma \max_a Q(s_{t+1}, a; \theta)$, from the Bellman equation. Following Mnih et al. [73], we make use of a (separate) target Q-network to calculate the expected Q-value, in order to have 'stable updates'. The target Q-network is periodically updated with the current parameters θ. We also make use of an experience replay memory $\mathcal{D}$ to store transitions. To perform updates, we sample a batch of transitions $(\hat{s}, \hat{a}, \hat{s}', r)$ at random from $\mathcal{D}$ and minimize the loss function[7]:

$$\mathcal{L}(\theta) \; = \mathrm{E}_{\hat{s},\hat{a}}[(Q(\hat{s}, \hat{a}; \theta) - y)^2]$$

where $y = r + \gamma \max_{a'} Q(\hat{s}', a'; \theta_t)$ is the target Q-value. The learning updates are made every training step using the following gradients:

$$\nabla_\theta \mathcal{L}(\theta) \; = \mathrm{E}_{\hat{s},\hat{a}}[2(Q(\hat{s}, \hat{a}; \theta) - y)\nabla_\theta Q(\hat{s}, \hat{a}; \theta)]$$

Algorithm 4.2 details the DQN training procedure.

[7]The expectation is over the transitions sampled uniformly at random from $\mathcal{D}$.

4.5 Experimental Setup

4.5.1 Data

We perform experiments on two different datasets. For the first set, we collect data from the Gun Violence archive,[8] a website tracking shootings in the United States. The data consists of one news article on each shooting and annotations for (1) the name of the shooter (*ShooterName*), (2) the number of people killed (*NumKilled*), (3) the number of people wounded (*NumWounded*), and (4) the city where the incident took place (*City*). We consider these to be our entities of interest, to be extracted from the articles. The second dataset we use is the Foodshield EMA database,[9] documenting adulteration incidents around the world since 1980. This data contains annotations for (1) the affected food product (*Food*), (2) the adulterant (*Adulterant*) and (3) the location of the incident (*Location*). Both datasets are classic examples where the number of recorded incidents is insufficient for large-scale IE systems to leverage successfully.

For each source article in the above databases, we download extra articles using the Bing Search API[10] with different automatically generated queries, limiting ourselves to the top 20 results per query. We use only the source articles from the *train* portion to learn the parameters of the base extractor. This is basically similar to how one would train a supervised model on the available data. To train the DQN agent (and the meta-classifier baseline, described in Section 4.5.4), we use the entire *train* set plus the downloaded articles. Note that the extra articles are noisy and most of them remain irrelevant to the original article. The parameters of all models are tuned on the *dev* set. For the final results, we train the models on the combined train and dev sets and use the entire *test* set (source + downloaded articles) to evaluate. Table 4.2 provides statistics on the data.

[8] `www.shootingtracker.com/Main_Page`
[9] `www.foodshield.org/member/login/`
[10] `www.bing.com/toolbox/bingsearchapi`

	Shootings			Adulteration		
	Train	**Test**	**Dev**	**Train**	**Test**	**Dev**
Source articles	306	292	66	292	148	42
Downloaded articles	8201	7904	1628	7686	5333	1537

Table 4.2: Number of articles in the *Shootings* and *Adulteration* datasets.

4.5.2 Base Extraction System

We use a maximum entropy classifier for the base extraction system, since it provides flexibility to capture various local context features and has been shown to perform well for information extraction [24]. The classifier is used to tag each word in a document as one of the entity types or not (e.g. {*ShooterName, NumKilled, NumWounded, City, Other*} in the *Shootings* domain). Then, for each tag except *Other*, we select the mode of the values in an article to obtain the set of entity extractions for that article.[11]

The types of features used in the classifier include:

- Word type indicator for each word in our vocabulary. This is set for the current word as well as words occurring within a context window of size three.
- Tag type indicator for the previous three words.
- Dictionary-based features for the current word plus context words within a neighborhood of three steps. A list of these features is provided in Table 4.3

Type	**Features**
Names	isMaleName, isFemaleName
Lexical	isCapital, isLongWord, isShortWord
Numeric	isDigit, containsDigit, isNumberWord, isOrdinalWord
Cities	isFullCity, isPartialCity

Table 4.3: Different types of dictionary-based binary features used in our Maximum Entropy classifier. These features are calculated both for the current word and words in the surrounding context.

[11] We normalize numerical words (e.g. "one" to "1") before taking the mode.

The features and context window $c = 4$ of neighboring words are tuned to maximize performance on a development set. We also experimented with a conditional random field (CRF) (with the same features) for the sequence tagging [26] but obtained worse empirical performance (see Section 4.6). The parameters of the base extraction model are not changed during training of the RL model.

4.5.3 Evaluation

We evaluate the extracted entity values against the gold annotations and report the corpus-level average accuracy on each entity type. For the ShooterName entity, the annotations (and the news articles) often contain multiple names (first and last) in various combinations, so we consider retrieving either name as a successful extraction. For all other entities, we look for exact matches.

4.5.4 Baselines

We explore 4 types of baselines:

- *Basic extractors:* The simplest baselines to consider are the base extraction systems themselves. Here, we make use of the CRF and the Maxent classifiers described previously.

- *Aggregation systems:* The second type of baselines we consider are basic aggregation systems. These methods first run the base extractor on both the original articles as well as the extra documents, and then utilize simple heuristics to aggregate the values. We examine two systems that perform different types of value reconciliation. The first model (*Confidence*) chooses the entity value for a slot by picking the one with the highest confidence score assigned by the base extractor. The second system (*Majority*) takes a majority vote over all values extracted from the different articles. Both methods filter new entity values using a threshold τ (tuned on the development set) on the cosine similarity over the tf-idf representations of the source and new articles.

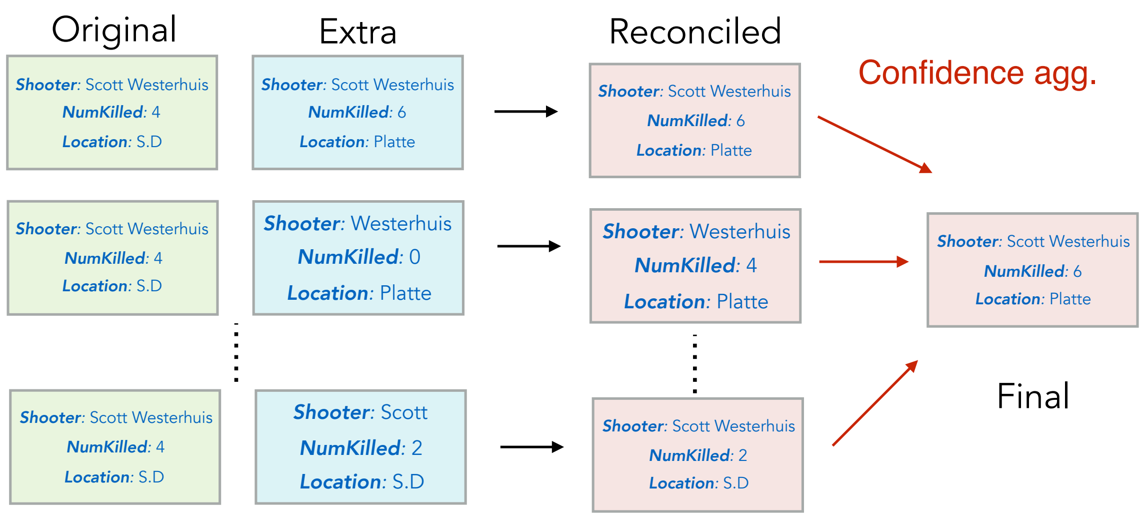


Figure 4-6: Schematic of the working of the meta-classifier. The base extractor is applied to both the original article and all extra articles downloaded using the corresponding queries. The meta-classifier takes a reconciliation decision for each pair of (original article, extra article) to produce sets of reconciled values. Finally, these values are aggregated using confidence scores to obtain the final set of values.

- *Meta-classifer:* To demonstrate the importance of modeling the problem in the RL framework, we consider a meta-classifier baseline. The classifier operates over the same input state space and produces the same set of reconciliation decisions $\{d\}$ as the DQN. For training, we use the original source article for each event along with a related downloaded article to compute the state. If the downloaded article has the correct value and the original one does not, we label it as a positive example for that entity class. If multiple such entity classes exist, we create several training instances with appropriate labels, and if none exist, we use the label corresponding to the *reject all* action. For each *test* event, the classifier is used to provide decisions for all the downloaded articles and the final extraction is performed by aggregating the value predictions using the *Confidence*-based scheme described above. Figure 4-6 provides a schematic depiction of the meta-classifier.

- *Oracle:* Finally, we also have an ORACLE score which is computed assuming perfect reconciliation and querying decisions on top of the Maxent base extractor. This helps us analyze the contribution of the RL system in isolation of the inherent limitations of the base extractor.

4.5.5 RL models

We perform experiments using three variants of RL agents:

1. *RL-Basic*: This variant performs only reconciliation decisions at every time step. Articles are presented to this agent in a round-robin fashion from the different query lists.

2. *RL-Query*: This agent takes only query decisions with the reconciliation strategy fixed (similar to Kanani and McCallum [54]). We use a confidence-based reconciliation, i.e. at each step, between two values, choose the one with the higher confidence assigned by the base extractor.

3. *RL-Extract*: This is our full system incorporating both reconciliation and query decisions.

These different models help us evaluate the relative importance of the reconciliation and query decisions. As we shall see in Section 4.6, the two types of actions are heavily interlinked; having the flexibility to choose both leads to substantially larger gains in performance.

4.5.6 Implementation details

We train the models for 10000 steps every epoch using the Maxent classifier as the base extractor. The final accuracies reported are averaged over 3 independent runs; each run's score is averaged over 20 epochs after 100 epochs of training. The penalty per step is set to -0.001. For the DQN, we use the dev set to tune all parameters. We used a replay memory $\mathcal{D}$ of size 500k, and a discount (γ) of 0.8. We set the learning rate to 2.5E^{-5}. The ϵ in ϵ-greedy exploration is annealed from 1 to 0.1 over 500k transitions. The target-Q network is updated every 5000 steps.

4.6 Results

4.6.1 Extraction Accuracy

Table 4.4 details the performance of all models on the two datasets. We observe that the base models, CRF and Maxent achieve somewhat low numbers, with the CRF in particular performing quite badly. The aggregation baselines, *Confidence* and *Majority* obtain slight improvements of 2-4%, but are unable to take full advantage of the extra articles available. The *meta-classifier* obtains similar performance to the aggregation baselines, in spite of being trained to reconcile values from the extra articles.

In contrast, our system (RL-Extract) obtains a substantial gain in accuracy over the basic extractors on all entity types in both domains. For instance, RL-Extract is 11.4% more accurate than the basic Maxent extractor on City and 7.1% better on NumKilled, while also achieving gains of more than 5% on the other entities on the *Shootings* domain. The gains on the *Adulteration* dataset are also significant, up to a 11.5% increase on the Location entity.

RL-Extract outperforms the aggregation baselines by 7.2% on *Shootings* and 5% on *Adulteration*, averaged over all entities. Further, the importance of sequential decision-making is demonstrated by RL-Extract performing significantly better than the meta-classifier (7.0% on *Shootings* over all entities). A key reason for this is the fact that the meta-classifier aggregates over the entire set of extra documents, including the long tail of noisy, irrelevant documents. RL-Extract, on the other hand, aggregates information from a small set of articles, tailored to the specific details sought.

From the same table, we also observe the advantage of providing our system with the flexibility of both query selection and value reconciliation. RL-Extract significantly outperforms both RL-Basic and RL-Query on both domains, emphasizing the need for jointly modeling the interplay between the two types of decisions.

Figure 4-7 shows the learning curve of the agent by measuring reward on the test

System	**Shootings**				**Adulteration**		
	ShooterName	**NumKilled**	**NumWounded**	**City**	**Food**	**Adulterant**	**Location**
CRF extractor	9.5	65.4	64.5	47.9	41.2	28.3	51.7
Maxent extractor	45.2	69.7	68.6	53.7	56.0	52.7	67.8
Confidence Agg. (τ)	45.2 (0.6)	70.3 (0.6)	72.3 (0.6)	55.8 (0.6)	56.0 (0.8)	54.0 (0.8)	69.2 (0.6)
Majority Agg. (τ)	47.6 (0.6)	69.1 (0.9)	68.6 (0.9)	54.7 (0.7)	56.7 (0.5)	50.6 (0.95)	72.0 (0.4)
Meta-classifier	45.2	70.7	68.4	55.3	55.4	52.7	72.0
RL-Basic	45.2	71.2	70.1	54.0	57.0	55.1	76.1
RL-Query (conf)	39.6	66.6	69.4	44.4	39.4	35.9	66.4
RL-Extract	**50.0**	**77.6***	**74.6***	**65.6***	**59.6***	**58.9***	**79.3***
ORACLE	57.1	86.4	83.3	71.8	64.8	60.8	83.9

Table 4.4: Accuracy of various baselines (italics), our models (RL-) and the ORACLE on *Shootings* and *Adulteration* datasets. **Agg.** refers to aggregation baselines. Bold indicates best system scores. *statistical significance of $p < 0.0005$ vs basic Maxent extractor using the Student-t test. Numbers in parentheses indicate the optimal threshold (τ) for the aggregation baselines. Confidence-based reconciliation was used for RL-Query.

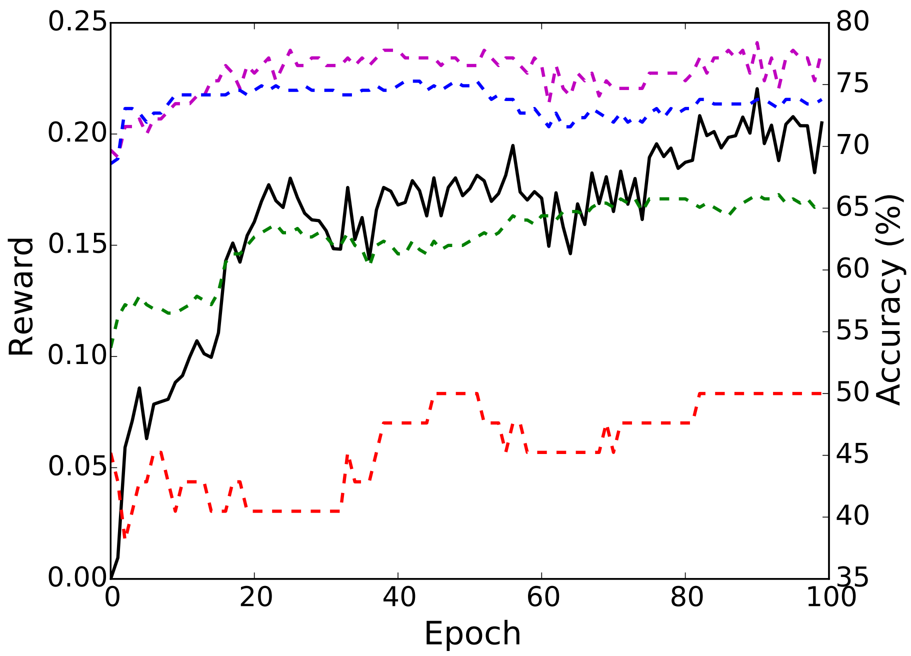


Figure 4-7: Evolution of average reward (solid black) and accuracy on various entities (dashed lines; red=*ShooterName*, magenta=*NumKilled*, blue=*NumWounded*, green=*City*) on the *test* set of the *Shootings* domain.

set after each training epoch. The reward improves gradually and the accuracy on each entity increases correspondingly. We do notice a trade-off in the accuracy of different entity values (around epoch 60), where the accuracy of *ShooterName* and *NumWounded* drop while that of *City* increases. However, this is quickly resolved with further learning and by epoch 100, all entity types reach peak accuracy.

4.6.2 Analysis

Table 4.5 provides some examples where our model extracts the right values although the base extractor fails to do so. In all of the cases, we see that the original article contains the correct information, albeit in a slightly complicated fashion for the base extractor. Our model is able to retrieve other articles with more prototypical language, resulting in successful extraction.

We also analyzed the importance of different reconciliation schemes, rewards and context-vectors in RL-Extract on the Shootings domain (Table 4.6). In addition to simple replacement (Replace), we experimented with using Confidence and Majority-

Entity	System: Value	Example
ShooterName	**Basic**: Stewart	A source tells Channel 2 Action News that Thomas Lee has been arrested in Mississippi ... Sgt . Stewart Smith, with the Troup County Sheriff's office, said.
	RL-Extract: Lee	Lee is accused of killing his wife, Christie; ...
NumKilled	**Basic**: 0	Shooting leaves 25 year old Pittsfield man dead , 4 injured
	RL-Extract: 1	One man is dead after a shooting Saturday night at the intersection of Dewey Avenue and Linden Street.
NumWounded	**Basic**: 0	Three people are dead and a fourth is in the hospital after a murder suicide
	RL-Extract: 1	3 dead, 1 injured in possible Fla. murder-suicide
City	**Basic**: Englewood	A 2 year old girl and four other people were wounded in a shooting in West Englewood Thursday night, police said
	RL-Extract: Chicago	At least 14 people were shot across Chicago between noon and 10:30 p.m. Thursday. The last shooting left five people wounded.

Table 4.5: Sample predictions (along with corresponding article snippets) on the *Shootings* domain. RL-Extract is able to produce correct values where the basic extractor (Maxent) fails, by retrieving alternative articles suited for easier extraction.

based reconciliation schemes for RL-Extract. We observe that the Replace scheme performs much better than the others (2-6% on all entities) and believe this is because it provides the agent with more flexibility in choosing the final values.

From the same table, we see that using the tf-idf counts of context words as part of the state provides better performance than using no context or using simple unigram counts. In terms of reward structure, providing rewards after each step is empirically found to be significantly better (>10% on average) compared to a single delayed reward per episode. The last column shows the average number of steps (≈ extra articles examined) per episode – the values range from 6.8 to 10.0 steps for the different schemes. The best system (RL-Extract with Replace, tf-idf and step-based rewards) uses 9.4 steps per episode.

Reconciliation (RL-Extract)	**Context**	**Reward**	**Accuracy**				**Steps**
			S	**K**	**W**	**C**	
Confidence	tf-idf	Step	47.5	71.5	70.4	60.1	8.4
Majority	tf-idf	Step	43.6	71.8	69.0	59.2	9.9
Replace	*No context*	Step	44.4	77.1	72.5	63.4	8.0
Replace	*Unigram*	Step	48.9	76.8	74.0	63.2	10.0
Replace	tf-idf	*Episode*	42.6	62.3	68.9	52.7	6.8
Replace	tf-idf	Step	**50.0**	**77.6**	**74.6**	**65.6**	9.4

Table 4.6: Effect of using different reconciliation schemes, context-vectors, and rewards in our RL framework (*Shootings* domain). The last row is the overall best scheme (deviations from this are in *italics*). Context refers to the type of word counts used in the state vector to represent entity context. Rewards are either per step or per episode. (**S**: ShooterName, **K**: NumKilled, **W**: NumWounded, **C**: City, **Steps**: Average number of steps per episode)

4.7 Conclusions

In this chapter, we have explored the task of acquiring and incorporating external evidence to improve information extraction accuracy for domains with limited access to training data. This process comprises issuing search queries, extraction from new sources and reconciliation of extracted values, repeated until sufficient evidence is obtained. We use a reinforcement learning framework and learn optimal action sequences to maximize extraction accuracy while penalizing extra effort. We show that our model, trained as a deep Q-network, outperforms traditional extractors by 7.2% and 5% on average on two different domains, respectively. We also demonstrate the importance of sequential decision-making by comparing our model to a meta-classifier operating on the same space, obtaining up to a 7% gain. We now summarize some follow-up work that has been performed and suggest future directions of research.

Future research

- The techniques presented in this chapter are not specific to information extraction. An autonomous system that can query over the vast web and aggregate information can prove useful to a variety of other NLP tasks such as open-domain question answering [90, 23] or automated dialogue systems [30, 38]. Future work can investigate such approaches in conjunction with improvements to base systems.

- A limitation in our current framework is that our choice of queries is restricted by our query templates. Recently proposed methods to reformulate queries on-the-fly [82, 17] can be use to improve the diversity of articles retrieved and provide greater accuracy gains.

- In our current setup, the base extraction model is trained once and its parameters remain fixed throughout the RL agent training phase. An interesting direction to explore is retraining the base model with the extra (noisy) data retrieved from the web.

- Another direction for improvement is in the training time for the RL agent. This enforces limits on the number of extra articles we can consider in our setup. Recent work [99] has explored replacing the DQN in our setup with an Actor Critic architecture [72]. They demonstrate a significant reduction in training speed without much loss in accuracy. Other RL algorithms could potentially provide further improvements in this direction.

Chapter 5

Conclusions

In this thesis, I have explored methods that integrate natural language understanding with autonomous behavior in interactive environments. Such frameworks enable a contextual interpretation of language semantics while alleviating dependence on large amounts of manual annotation. As we demonstrate through several experiments, our models can exploit unstructured feedback in the form of reward signals to learn meaningful representations for text, which are optimized for utilization in the end task. Further, we also demonstrate that leveraging the knowledge encoded in text can provide significant boosts to performance of autonomous agents in control applications. Textual information provides connections between the dynamics of the environment that may otherwise require a significant number of interactions to discover.

I have demonstrated these ideas in the context of three different applications. First, we looked at an approach for learning to play text-based games, where all interaction is through natural language. Second, I exhibited a framework for utilizing textual descriptions to assist cross-domain policy transfer for reinforcement learning. Finally, I showed how reinforcement learning can enhance traditional NLP systems for tasks such as information extraction in low resource scenarios. Together, these methods open up new research directions, towards a tighter coupling between semantic interpretation of language and control policies of artificially intelligent agents.

Future directions

We now provide some future directions of research that follow from this thesis:

- **Integrate visual understanding**: Visual stimuli are a major sensory component of our lives, and it is imperative for an AI system to demonstrate visual perception and understanding. Vision provides more context for language and policies, while meaningful representations can be learned using linguistic knowledge and feedback from control applications. Although recent work [48, 22] has started to show promise in this area, further research is needed to form a greater synergy between these components, a key requirement for achieving general artificial intelligence.

- **Leverage generic knowledge** Another research direction is to leverage information present in knowledge bases such as Freebase [13] to improve performance on control applications. Such knowledge is not directly aligned with the end task, and leveraging the most relevant information is a major challenge. However, systems can still benefit from the extra information to improve their control policies.

- **Explore hierarchical control policies**: Information can be present in text at multiple levels of abstraction. In this thesis, we make no explicit distinction between these while integrating with control applications. One direction for future research could be to integrate hierarchical methods for reinforcement learning [60] with semantic interpretation, and thereby explicitly leveraging text at different granularities.

Appendix A

Language Understanding for Text-based Games

A.1 Environment details

A.1.1 Rewards

Table A.1 shows the reward structure used in our games. Positive rewards are provided for completing quests and reaching significant checkpoints in the game. Negative rewards are provided for invalid commands and failures.

World	**Positive**	**Negative**
Home	Quest goal: +1	Negative per step: -0.01
		Invalid command: -0.1
Fantasy	Cross bridge: +5	Fall from bridge: -0.5
	Defeat guardian: +5	Lose to guardian: -0.5
	Reach tomb: +10	Negative per step: -0.01
		Invalid command: -1

Table A.1: Reward structure used in the two game worlds.

A.1.2 Home World

A schematic of the Home world is provided in Figure A-1. The world consists of four rooms, each containing a characteristic object. A player is randomly placed in one of the rooms and provided a quest.

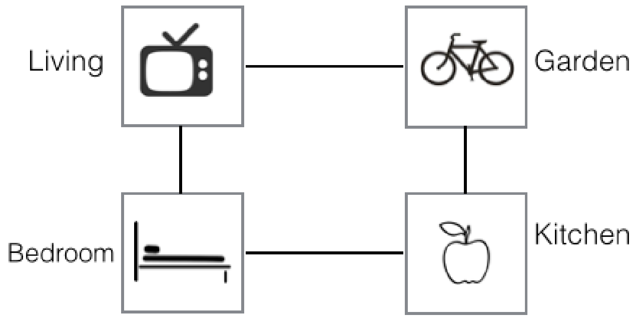


Figure A-1: Rooms and objects in the Home World with connecting pathways.

Appendix B

Policy Transfer via Language Grounding

We provide a few sample descriptions for each domain used in our experiments. These sentences were collected by asking workers on Amazon Mechanical Turk to watch videos of gameplay and describe each entity observed on screen.

B.1 Example text descriptions

B.1.1 Freeway

- Car1: *vehicle that is red and fast going right*
- Car2: *car that is going slow and to the right*
- Car4: *enemy that moves horizontally to the left quickly*
- Tree1: *these items do not move but also block the player from moving to their space*

Figure B-1: Example text descriptions of entities in Freeway.

B.1.2 Friends and Enemies

- Scorpion2: *enemy that moves to the north*
- Bear2: *enemy who chases you*
- Bee1: *black bug that can be collected by the player*
- Bird2: *a bird that must be avoided*

Figure B-2: Example text descriptions of entities in Friends and Enemies (F&E).

B.1.3 Bomberman

- Flame1: *this fires from bombs and will destroy anything it touches*
- Spider3:*enemy that moves horizontally left and right*
- Scorpion4: *enemy that moves randomly*
- Scorpion4: *a fairly slow moving enemy*

Figure B-3: Example text descriptions of entities in Bomberman.

B.1.4 Boulderchase

- Bat1: *enemy that can dig through dirt*
- Pickaxe1: *pickaxe that can be used to dig the earth*
- Scorpion3: *enemy that follows player*
- Diamond1: *this item is picked up by the player when they move across it*

Figure B-4: Example text descriptions of entities in Boulderchase.

B.2 Reward curves

Below, we provide the reward curves for the transfer conditions of F&E-1 → F&E-2 and Bomberman → Boulderchase.

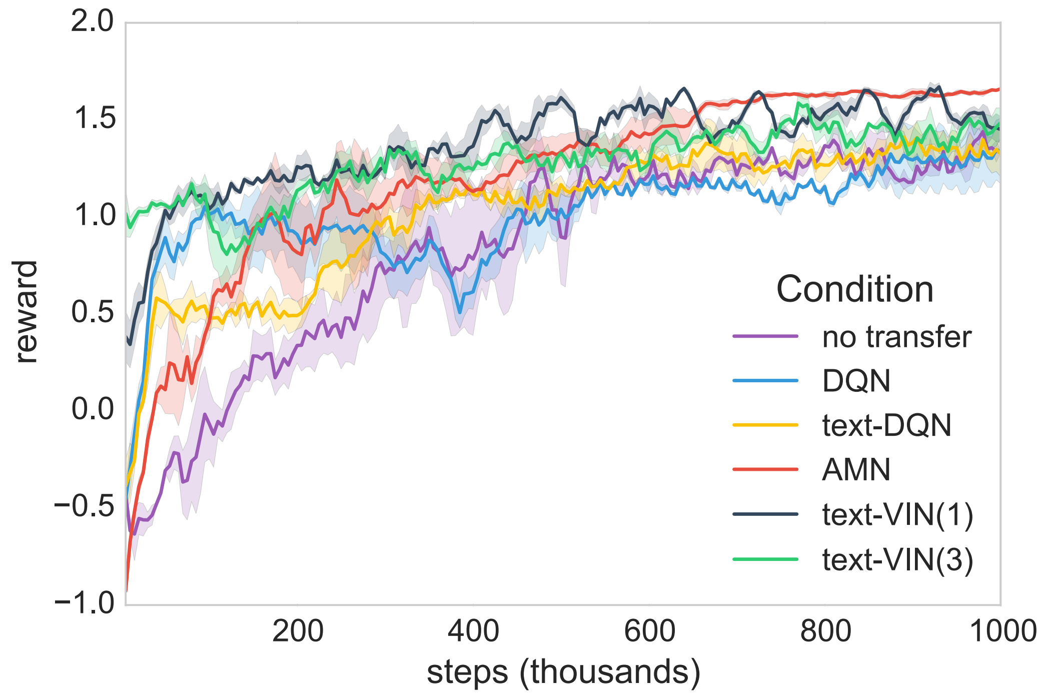


Figure B-5: Reward curve for transfer condition F&E-1 → F&E-2. Numbers in parentheses for TEXT-VIN indicate k value. All graphs averaged over 3 runs with different seeds; shaded areas represent bootstrapped confidence intervals.

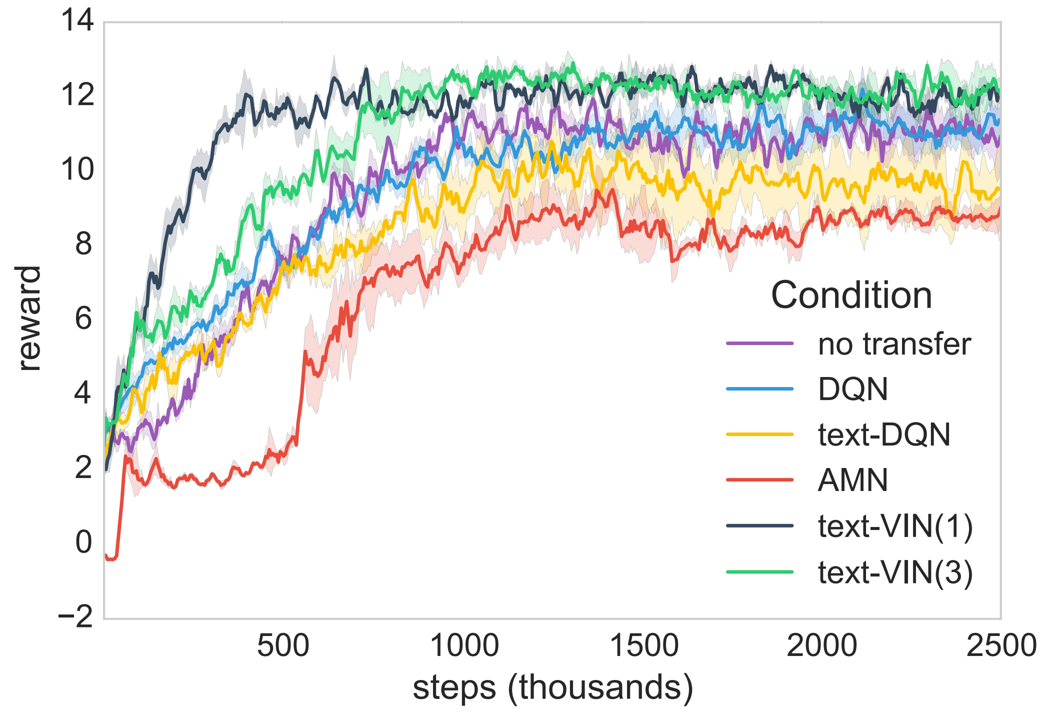


Figure B-6: Reward curve for transfer condition Bomberman → Boulderchase. Numbers in parentheses for TEXT-VIN indicate k value. All graphs averaged over 3 runs with different seeds; shaded areas represent bootstrapped confidence intervals.

Appendix C

Improved Information Extraction with Reinforcement Learning

C.1 Query templates

Here, we provide examples of query templates used for the domain of food adulteration incidents, sourced from the EMA database. The query templates were created automatically, using high co-occurence words with gold values in the training set.

$\langle$*title*$\rangle$
$\langle$*title*$\rangle$ + (state | country | india | china | province)
$\langle$*title*$\rangle$ + (adulterated | fake | food | products | samples)
$\langle$*title*$\rangle$ + (food | oil | milk | honey | price | brands | formula)

Table C.1: Examples of different query templates used by our model in web searches for articles on food adulteration. The | symbol represents logical OR. The last three queries contain context words around values for entity types Location, adulterant and food, respectively. At query time, $\langle$*title*$\rangle$ is replaced by the source article's title.

Bibliography

[1] Charu C Aggarwal and ChengXiang Zhai. *Mining text data*. Springer Science & Business Media, 2012.

[2] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94. ACM, 2000.

[3] Christopher Amato and Guy Shani. High-level reinforcement learning in strategy games. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1*, pages 75–82. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[4] Eyal Amir and Patrick Doyle. Adventure games: A challenge for cognitive robotics. In *Proc. Int. Cognitive Robotics Workshop*, pages 148–155, 2002.

[5] Jacob Andreas and Dan Klein. Alignment-based compositional semantics for instruction following. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015.

[6] Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62, 2013.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[8] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation (amr) 1.0 specification. In *Parsing on Freebase from Question-Answer Pairs. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle: ACL*, pages 1533–1544, 2012.

[9] Michele Banko, Eric Brill, Susan Dumais, and Jimmy Lin. Askmsr: Question answering using the worldwide web. In *Proceedings of 2002 AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*, pages 7–9, 2002.

[10] Cosmin Adrian Bejan and Sanda Harabagiu. Unsupervised event coreference resolution. *Computational Linguistics*, 40(2):311–347, 2014.

[11] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, volume 2, page 6, 2013.

[12] Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

[13] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.

[14] SRK Branavan, David Silver, and Regina Barzilay. Learning to win by reading manuals in a monte-carlo framework. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 268–277. Association for Computational Linguistics, 2011.

[15] SRK Branavan, David Silver, and Regina Barzilay. Non-linear monte-carlo search in Civilization II. AAAI Press/International Joint Conferences on Artificial Intelligence, 2011.

[16] SRK Branavan, Luke S Zettlemoyer, and Regina Barzilay. Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1268–1277. Association for Computational Linguistics, 2010.

[17] Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. Ask the right questions: Active question reformulation with reinforcement learning. *arXiv preprint arXiv:1705.07830*, 2017.

[18] Michael Buhrmester, Tracy Kwang, and Samuel D. Gosling. Amazon's mechanical turk. *Perspectives on Psychological Science*, 6(1):3–5, 2011. PMID: 26162106.

[19] Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110. ACM, 2010.

[20] Angel X Chang, Valentin I Spitkovsky, Eric Yeh, Christopher D Manning, and Eneko Agirre. Stanford-ubc entity linking at tac-kbp. In *TAC*, 2010.

[21] Chia-Hui Chang, Mohammed Kayed, Moheb R Girgis, and Khaled F Shaalan. A survey of web information extraction systems. *IEEE transactions on knowledge and data engineering*, 18(10):1411–1428, 2006.

[22] Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. *arXiv preprint arXiv:1706.07230*, 2017.

[23] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[24] Hai Leong Chieu and Hwee Tou Ng. A maximum entropy approach to information extraction from semi-structured and free text. In *Proceedings of AAAI*, 2002.

[25] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of the joint SIGDAT conference on empirical methods in natural language processing and very large corpora*, pages 100–110, 1999.

[26] Aron Culotta and Andrew McCallum. Confidence estimation for information extraction. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 109–112. Association for Computational Linguistics, 2004.

[27] Pavel Curtis. Mudding: Social phenomena in text-based virtual realities. *High noon on the electronic frontier: Conceptual issues in cyberspace*, pages 347–374, 1992.

[28] Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8599–8603. IEEE, 2013.

[29] Mark A DePristo and Robert Zubek. being-in-the-world. In *Proceedings of the 2001 AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, pages 31–34, 2001.

[30] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–495, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[31] Yunshu Du, V Gabriel, James Irwin, and Matthew E Taylor. Initial progress in transfer for deep reinforcement learning algorithms. In *Proceedings of Deep Reinforcement Learning: Frontiers and Challenges Workshop, New York City, NY, USA*, 2016.

[32] Greg Durrett and Dan Klein. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490, 2014.

[33] Jacob Eisenstein, James Clarke, Dan Goldwasser, and Dan Roth. Reading to learn: Constructing features from semantic abstracts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 958–967, Singapore, August 2009. Association for Computational Linguistics.

[34] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. Open information extraction: The second generation. In *IJCAI*, volume 11, pages 3–10, 2011.

[35] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics, 2011.

[36] Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 397–406, Doha, Qatar, October 2014. Association for Computational Linguistics.

[37] Milica Gašic, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. Pomdp-based dialogue manager adaptation to extended domains. In *Proceedings of SIGDIAL*, 2013.

[38] Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. A knowledge-grounded neural conversation model. *arXiv preprint arXiv:1702.01932*, 2017.

[39] Ruben Glatt and Anna Helena Reali Costa. Policy reuse in deep reinforcement learning. In *AAAI*, pages 4929–4930, 2017.

[40] Peter Gorniak and Deb Roy. Speaking with your sidekick: Understanding situated speech in computer role playing games. In R. Michael Young and John E. Laird, editors, *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference, June 1-5, 2005, Marina del Rey, California, USA*, pages 57–62. AAAI Press, 2005.

[41] Ralph Grishman. Information extraction. In *The Oxford handbook of computational linguistics*. Oxford University Press, 2012.

[42] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017.

[43] Kelvin Guu, John Miller, and Percy Liang. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 318–327, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[44] Xianpei Han, Le Sun, and Jun Zhao. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 765–774. ACM, 2011.

[45] Brent Harrison, Upol Ehsan, and Mark O Riedl. Guiding reinforcement learning exploration using natural language. *arXiv preprint arXiv:1707.08616*, 2017.

[46] Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630, Berlin, Germany, August 2016. Association for Computational Linguistics.

[47] Sachithra Hemachandra, Matthew R Walter, Stefanie Tellex, and Seth Teller. Learning spatial-semantic representations from natural language descriptions and scene classifications. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2623–2630. IEEE, 2014.

[48] Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojtek Czarnecki, Max Jaderberg, Denis Teplyashin, et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017.

[49] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[50] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[51] Mohit Iyyer, Jordan L Boyd-Graber, Leonardo Max Batista Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *EMNLP*, pages 633–644, 2014.

[52] Heng Ji and Ralph Grishman. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1148–1158, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[53] Rosie Jones. *Learning to extract entities from labeled and unlabeled text.* PhD thesis, Carnegie Mellon University, Language Technologies Institute, School of Computer Science, 2005.

[54] Pallika H Kanani and Andrew K McCallum. Selecting actions for resource-bounded information extraction using reinforcement learning. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 253–262. ACM, 2012.

[55] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[56] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. In *Human-Robot Interaction (HRI), 2010 5th ACM/IEEE International Conference on*, pages 259–266. IEEE, 2010.

[57] George Konidaris and Andrew G Barto. Building portable options: Skill transfer in reinforcement learning. In *IJCAI*, volume 7, pages 895–900, 2007.

[58] George D Konidaris. A framework for transfer in reinforcement learning. In *ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

[59] Jan Koutník, Giuseppe Cuccu, Jürgen Schmidhuber, and Faustino Gomez. Evolving large-scale neural networks for vision-based reinforcement learning. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 1061–1068. ACM, 2013.

[60] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 3675–3683, 2016.

[61] Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. Learning to automatically solve algebra word problems. *ACL (1)*, pages 271–281, 2014.

[62] Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. In *2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*. Association for Computational Linguistics (ACL), 2013.

[63] Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 489–500, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[64] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[65] Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*, 2015.

[66] Yaxin Liu and Peter Stone. Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the national conference on artificial intelligence*, volume 21, page 415. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[67] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[68] Gideon S Mann and David Yarowsky. Multi-field information extraction and cross-document fusion. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 483–490. Association for Computational Linguistics, 2005.

[69] Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. Learning to parse natural language commands to a robot control system. In *Experimental Robotics*, pages 403–415. Springer, 2013.

[70] Andrew McCallum. Information extraction: Distilling structured data from unstructured text. *Queue*, 3(9):48–57, 2005.

[71] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[72] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

[73] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and

Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015.

[74] Andrew W Moore and Christopher G Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130, 1993.

[75] Ion Muslea et al. Extraction patterns for information extraction tasks: A survey. In *The AAAI-99 workshop on machine learning for information extraction*, volume 2. Orlando Florida, 1999.

[76] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. 2015.

[77] Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. Deep transfer in reinforcement learning by language grounding. *arXiv preprint arXiv:1708.00133*, 2017.

[78] Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015.

[79] Karthik Narasimhan, Adam Yala, and Regina Barzilay. Improving information extraction by acquiring external evidence with reinforcement learning. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[80] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. Compositional vector space models for knowledge base completion. 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 156–166, Beijing, China, July 2015. Association for Computational Linguistics.

[81] Trung Nguyen, Tomi Silander, and Tze Y Leong. Transferring expectations in model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2555–2563, 2012.

[82] Rodrigo Nogueira and Kyunghyun Cho. Task-oriented query reformulation with reinforcement learning. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2017.

[83] Martha Palmer, Daniel Gildea, and Nianwen Xue. Semantic role labeling. *Synthesis Lectures on Human Language Technologies*, 3(1):1–103, 2010.

[84] Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. Memen: Multi-layer embedding with memory networks for machine comprehension. *arXiv preprint arXiv:1707.09098*, 2017.

[85] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *International Conference on Learning Representations*, 2016.

[86] Fuchun Peng and Andrew McCallum. Information extraction from research papers using conditional random fields. *Information processing & management*, 42(4):963–979, 2006.

[87] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12, 2014.

[88] Diego Perez-Liebana, Spyridon Samothrakis, Julian Togelius, Tom Schaul, and Simon M Lucas. General video game ai: Competition, challenges and opportunities. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[89] Hoifung Poon and Pedro Domingos. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 1–10. Association for Computational Linguistics, 2009.

[90] John Prager et al. Open-domain question–answering. *Foundations and Trends® in Information Retrieval*, 1(2):91–231, 2007.

[91] Janarthanan Rajendran, Aravind Lakshminarayanan, Mitesh M Khapra, Balaraman Ravindran, et al. *a2t*: Attend, adapt and transfer: Attentive deep architecture for adaptive transfer from multiple sources. 2017.

[92] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[93] Himanshu Sahni, Saurabh Kumar, Farhan Tejani, Yannick Schroecker, and Charles Isbell. State space decomposition and subgoal creation for transfer in deep reinforcement learning. *3rd Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM 2017)*, 2017.

[94] Sunita Sarawagi and William W Cohen. Semi-markov conditional random fields for information extraction. In *Advances in neural information processing systems*, pages 1185–1192, 2005.

[95] Sunita Sarawagi et al. Information extraction. *Foundations and Trends® in Databases*, 1(3):261–377, 2008.

[96] Tom Schaul. A video game description language for model-based or interactive learning. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pages 1–8. IEEE, 2013.

[97] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1312–1320, 2015.

[98] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[99] Aditya Sharma, Zarana Parekh, and Partha Talukdar. Speeding up reinforcement learning-based information extraction training using asynchronous methods. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2017.

[100] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[101] David Silver, Richard S Sutton, and Martin Müller. Reinforcement learning of local shape in the game of go. In *IJCAI*, volume 7, pages 1053–1058, 2007.

[102] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934, 2013.

[103] Mihai Surdeanu, David McClosky, Julie Tibshirani, John Bauer, Angel X Chang, Valentin I Spitkovsky, and Christopher D Manning. A simple distant supervision approach for the tac-kbp slot filling task. In *Proceedings of Text Analysis Conference 2010 Workshop*, 2010.

[104] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[105] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT Press, 1998.

[106] István Szita. Reinforcement learning in games. In *Reinforcement Learning*, pages 539–577. Springer, 2012.

[107] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. 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 1556–1566, Beijing, China, July 2015. Association for Computational Linguistics.

[108] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, pages 2154–2162, 2016.

[109] Matthew E Taylor, Nicholas K Jong, and Peter Stone. Transferring instances for model-based reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 488–505. Springer, 2008.

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

[111] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4, 2012.

[112] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*, 2017.

[113] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[114] Adam Vogel and Dan Jurafsky. Learning to follow navigational directions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 806–814. Association for Computational Linguistics, 2010.

[115] Matthew R Walter, Sachithra Hemachandra, Bianca Homberg, Stefanie Tellex, and Seth Teller. Learning semantic maps from natural language descriptions. Robotics: Science and Systems, 2013.

[116] Zhuoran Wang, Tsung-Hsien Wen, Pei-Hao Su, and Yannis Stylianou. Learning domain-independent dialogue policies via ontology parameterisation. In *SIGDIAL Conference*, pages 412–416, 2015.

[117] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[118] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*, pages 515–526. ACM, 2014.

[119] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[120] Fei Wu and Daniel S Weld. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127. Association for Computational Linguistics, 2010.

[121] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*, 2016.

[122] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[123] Haiyan Yin and Sinno Jialin Pan. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 1640–1646, 2017.