Deep Reinforcement Learning in Natural Language Scenarios

Ji He

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2017

Reading Committee:

Mari Ostendorf, Chair

Xiaodong He

Luke Zettlemoyer

Program Authorized to Offer Degree:
Electrical Engineering

University of Washington

Abstract

Deep Reinforcement Learning in Natural Language Scenarios

Ji He

Chair of the Supervisory Committee:
Professor Mari Ostendorf
Electrical Engineering

Reinforcement learning refers to a class of algorithms that aim at learning a good policy in a dynamic environment. Recently, by combining deep learning with reinforcement learning, researchers have made significant breakthroughs in many artificial intelligence applications. The most notable applications are Atari games and game of Go. However, natural language applications involving deep reinforcement learning are still rare.

This thesis studies deep reinforcement learning in natural language scenarios with three contributions. First we introduce a novel architecture for reinforcement learning with deep neural networks designed to handle state and action spaces characterized by natural language. The architecture represents state and action spaces with separate embedding vectors, which are combined with an interaction function to approximate the Q-function in reinforcement learning. Second, we investigate reinforcement learning with a combinatorial, natural language action space. Novel deep reinforcement learning architectures are studied for effective modeling of the value function associated with actions comprised of interdependent sub-actions, accounting for redundancy among sub-actions. In addition, a two-stage Q-learning framework is introduced as a strategy for reducing the cost to search the combinatorial action space. Third, we augment the state representation to incorporate global context using an external unstructured knowledge source with temporal information. This approach is inspired by the observation that in a real-world decision making process, it is

usually beneficial to consider background knowledge and popular current events relevant to the current local context.

We experiment on two types of tasks, text-based games and predicting popular Reddit discussion threads. We show that all contributions help reinforcement learning in natural language scenarios. Specifically, experiments with paraphrased action descriptions on text games show that separate modeling of state and action spaces is extracting meaning rather than simply memorizing strings of text. For a combinatorial action space, our proposed model, which represents dependence between sub-actions through a bi-directional LSTM, gives the best performance for predicting popular Reddit threads across different domains. The two-stage Q-learning achieves significant performance gain compared to random sampling a subspace of the combinatorial action space. For tracking the most popular thread, incorporating external knowledge in the form of discussions about world news also leads to significant improvements with a 34% gain for discussions about topic (politics) for which world news is particularly relevant.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Many people have helped me towards my PhD degree, starting with the people who suggested me to pursue a PhD back in my undergraduate program, or even earlier in high school, where I established my respect for rigorous theory and perseverance to tolerate this long and arduous journey.

First and foremost, I would like to express my sincere thanks to my supervisor, Prof. Mari Ostendorf. She has always been supportive, even during the first few years when I really struggled and made little progress. By the time I need to decide my thesis direction, we had multiple topics on the table, and she encouraged me to take this one. Looking back the decision was very visionary, as deep reinforcement learning had just started to thrive and show its impact, and we became some of the pioneers in applying deep reinforcement learning to the natural language processing area. On the other hand, this topic fits with my academic background and I am also fortunate to have pursued a direction I have been passionate about even until today. Working with Prof. Mari Ostendorf, there were happy times and there were tough times, but there was no bad time.

Second, I would like to thank Dr. Xiaodong He, who was my mentor during my internship at Microsoft Research, and then became my committee member in my fourth and fifth years. I learned much about deep learning from Xiaodong, and reinforcement learning from my internship, so that I could later on pick up recent advances on my own. Xiaodong also provided very detailed instructions on experimental designs and shaped my research taste.

I would also like to give special thanks to Prof. Michael Perlman, who is my GSR (Graduate School Representative) and introduced me to a concurrent Master's program in department of statistics. I did well in both his statistical inference courses, and also studied

comprehensively in statistical learning and stochastic modeling. These equip me with a solid mathematical background for research, as well as additional advantage when seeking jobs.

I owe many thanks to the rest of my committee. Both Prof. Hannaneh Hajishirzi and Prof. Luke Zettlemoyer provided insightful comments during my general exam and early stage of my thesis work. I benefited a lot from Microsoft researchers, such as Li Deng, Lihong Li, Jianshu Chen, and Jianfeng Gao. They as well as Mari provided advising help during comprising of the large scale reinforcement learning task. Labmates from the SSLI lab, as well as the TIAL lab later on, are my best friends, and because everyone has different specialty, I was exposed to different techniques overall. The list includes but is not limited to Amittai Axelrod, Sangyun Hahn, Brian Hutchinson, Yuzong Liu, Alex Marin, Julie Medero, Nicole Nichols, Bin Zhang, Hao Cheng, Hao Fang, Aaron Jaech, Kevin Lybarger, Yi Luan, Farah Nadeem, Trang Tran, and Victoria Zayats.

Part of the work in my dissertation was funded by the DARPA DEFT project. I was also allowed to explore freely and pursue my thesis direction while interning at Microsoft Research. This work could not have been done without their support.

One unique aspect of the PhD experience is that I got to know other peers (PhD students) in my field of study, from all around the world. Some of these connections were gained by attending conferences, while some might be through reading their papers, personal webpages, or social networks. I owe a lot to communicating with them, and their perseverance and work ethics really inspire me to try to not be left behind.

Finally, I would like to express my thanks to my parents. Although they don't know much about my research, they try to understand what it means to pursue a PhD. They became less involved in my education as time went by, but I learned most of work ethics from them.

Chapter 1

INTRODUCTION

1.1 Long-term decision making in natural language scenarios

Many artificial intelligence tasks involve sequential decision making and delayed rewards, such as video gaming, human-computer dialogue systems, newsfeed recommendation, and strategic financial/business planning. Reinforcement learning is a collection of algorithms for automatically learning policies to interact with an environment. The goal is to learn an optimal policy under which, the accumulated rewards will be maximized, by taking a specific action sequence (based on states given by the environment). At each time step, the agent (e.g. computer program) can automatically evaluate feasible actions and select one, and this action will affect the future states. The set of all possible actions is called the action space.

Researchers have been successfully applying reinforcement learning for various applications with a constrained action space. For previous reinforcement learning applications, actions are defined through task-specific features, or simple combinations of these features. For example, in Atari games, an action is carried out through a controller, where a joystick and a button is provided. The state is the image presented to the player. At each time step, different directions of the joystick and whether or not to push the button form a total of 18 feasible actions, and these 18 actions form the entire action space throughout the gameplay. In the game of Go, the state includes the stone color (player stone, opponent stone or empty), how many turns since a move was played, number of empty adjacent points, etc. The action is a valid move for the current player, which is no more than the total number of points on the board. In an America put option, the holder has the right to sell stock at a pre-determined strike price on any day before the expiration date. The state is the stock price, which is a continuous value, and the action is either to wait (till the next day) or to

exercise the option.

This thesis is about reinforcement learning in natural language scenarios. That is, the state and the action descriptions about the environment are in the form of natural language text. Since language is the most natural way to communicate with people, and to build an automated agent helping with human productivity, it is often desired that an agent is able to understand and execute some instructions in natural language. For example, in human-computer or human-robot dialog systems, the agent needs to understand the dialog state, which is usually the history of the conversation in the current dialog session. The state is thus large and sparse, since there are numerous ways people talk about things. The dialog manager will need to base its actions upon correct understanding of the state.

There are also situations where actions are defined through natural language. One example is text games, where sometimes a player (human or computer agent) chooses an action by clicking on a string of text (could be a phrase or a natural language sentence), and the story will be directed to different developments, with different rewards corresponding to good/bad endings. Another example is a human-computer dialog system, where the action is the response generated by the dialog manager.

Text-based actions are also involved in tracking news feeds or popular discussion threads in social media, where a computer agent intends to recommend popular commentary for a user to read. Here, the goal is for the recommendation system to identify and track written documents (e.g. news articles, comments in discussion forum threads, or scientific articles) in real time – attempting to identify hot updates before they become hot to keep the reader at the leading edge. The premise is that the user's bandwidth is limited, and only a limited number of things can be recommended out of several possibilities. At each time step, the agent is presented with new comments associated with each tracked thread, and the agent takes an action by choosing which of the new comments to track. A subset is formed from tracked threads, and rewards can be assigned by reader popularity votes.

Actions in reinforcement learning that are defined through natural language can involve a large vocabulary. Because a piece of natural language text can have arbitrary length,

the potential action space in these problems is unbounded. Furthermore, the feasible set of actions at different time steps can vary (e.g. the number of links presented to the player, the number of new comments to be tracked). Thus, a fixed-output architecture such as that used in an Atari game cannot handle these applications.

Another challenge with natural language based actions is language understanding in context. The agent has to choose the action with the most important/relevant text, given the current state, where the importance/relevance of a text string is measured by the accumulated future rewards. For example, text games often involve language understanding of the story and pragmatic clues under each action, while tracking popular discussion threads requires the agent to pick the comments that contribute the most to the current discussion topics. In this thesis, we address these problems by developing novel architectures for deep reinforcement learning.

1.2 Experimental Tasks

Two very different tasks are explored in order to experimentally assess our contributions. The first is text based games. Recently, video games such as Atari games are widely studied using deep reinforcement learning. In natural language processing, a text game is a natural counterpart to video games in computer vision. However, the scale of text games and size of vocabulary in one game are relatively small, which motivates us to propose a second large scale task, tracking and predicting popular discussion threads. Similar to newsfeed recommendation, in this task the agent tries to pick a number of discussion threads with high popularity to the reader. Because online users respond to previous comments in real time, this dynamic setting makes reinforcement learning a natural formulation, and millions of comments make it easier to apply sophisticated neural architectures.

Text games, although simple compared to video games, still enjoy high popularity in world-wide online communities.[1] There are annual competitions[2] held online since 1995 and

[1] http://www.intfiction.org/forum/, http://ifdb.tads.org/

[2] http://www.ifcomp.org/

has received hundreds of submissions so far. Text games communicate to players in the form of text display, and players understand the revealed texts and respond by typing or clicking text [1]. That is, the games have a high requirement for the text understanding ability of the agent.

Text games are complex due to two major reasons. First, they often involve language understanding of the story and pragmatic clues under each action. Players usually have to combine both the story and choices to infer the appropriate actions (e.g. Given "In front there is a lion", and action "go ahead", the player is more likely to die). The second reason is long term dependency. A player's early action might influence the later-on story development, as in the example of finding a key to unlock an object that is encountered later. Because a player's behavior (policy) changes how the environment interacts with him or her, reinforcement learning is appropriate for modeling long-term dependency in text games.

Text-based games represent a good starting point for exploring reinforcement learning, but current games do not have the richness in language represented by an open vocabulary. In order to explore a more challenging task, we also look at comment recommendation in social media.

In this thesis, we consider Reddit popularity prediction, which is similar to newsfeed recommendation but different in two respects. First, our goal is not to make recommendations based on an individual's preferences, but instead based on the anticipated long-term interest level of a broad group of readers from a target community. Second, we try to predict rather than detect popularity. Unlike individual interests, community interest level is not often immediately clear; there is a time lag before the level of interest starts to take off. In our experimental work, we use discussion forum text, where the recommendations correspond to recent posts or comments, assessing interest based on community response as observed in "likes" or other positive reactions to those comments. For training purposes, we can use community response measured at a time much later than the original post or publication. This problem is well-suited to the reinforcement learning paradigm, since the reward (the level of community uptake or positive response) is not immediately known, so the system needs

to learn a mechanism for estimating future reactions. Different from typical reinforcement learning, the action space is combinatorial since an action corresponds to a set of comments (sub-actions) chosen from a larger set of candidates.

Thread popularity tracking can be thought of as a proxy task for news or scientific article recommendation. It has the advantages that "documents" (comments) are relatively short and that the long-term reward can be characterized by Reddit voting scores, which makes this task easier to work with for algorithm development than these larger related tasks.

The comment tracking tasks introduce two challenges to reinforcement learning including the development of a framework for estimating the long-term reward (the Q-value in reinforcement learning) from a combination of sub-actions characterized by natural language, and the potentially high computational complexity of the combinatorial action space.

1.3 Approach

We address these challenges using value-based reinforcement learning. More specifically, we assume the environment is a Markov decision process (in contrast to a partially observable Markov decision process) and focus directly on estimating action value function. Because a natural language scenario presents both large state and action spaces, we use deep neural networks to transform the state and each action into continuous space representations. Our major contributions are novel neural architectures for modeling Q-values with natural language actions. We build agents that model the state space and action space separately, and using a general interaction function to predict the expected accumulated future rewards by taking a particular action.

We further investigate the problem of a combinatorial action space in two respects: modeling dependency between sub-actions and addressing complexity of searching a large combinatorial action space. We address the first problem by introducing a bi-directional LSTM deep neural network architectures to account for the potential redundancy and/or temporal dependency of different sub-actions in relation to the state space. To reduce computational complexity, we propose a two-stage Q-learning approach in a coarse-to-fine fashion, similar

to a beam search, which gives much better performance compared to random sampling a subset of actions in a large combinatorial action space.

A major distinction between decision making in a natural language scenario and a domain specific reinforcement learning task is that incorporating external knowledge usually helps. We propose using an attention mechanism based on temporal features, semantic similarity and popularity for enriching the state representation with a global context, and show that this significantly improves performance of the original reinforcement learning task. We also explore the use of temporal context in the Reddit popularity prediction task, specifically timing features associated with comments, and show that they give only small gains in performance.

Besides algorithmic development, the work makes contributions to research infrastructure. We release source codes for simulators on github.[3] Unlike in computer vision (e.g. Atari games) or robotics control (e.g. mountain car), there are not many publicly available testbeds related to reinforcement learning in natural language processing. For most researchers in natural language processing, reinforcement learning is relatively new. Thus proposing publicly available testbeds is especially important for the community.

1.4 Dissertation Overview and Contributions

The rest of this dissertation is organized as follows: Chapter 2 describes related background for understanding reinforcement learning, especially the recent development of deep reinforcement learning. Since our approach models natural language text using deep learning, a section on continuous space language processing is also included.

Chapter 3 introduces the two tasks studied in this thesis with more details, and shows how those problems can be formulated using reinforcement learning. Specifically, for text games we study choice-based and hypertext-based games. The action space in parser-based games, in contrast, have more constrained structures. For predicting popular Reddit threads,

[3]https://github.com/jvking/text-games, https://github.com/jvking/reddit-RL-simulator

we present a motivating example to show the environment is dynamic and has long-term dependency. We further explain how the task could lead to a combinatorial action space.

In Chapter 4 we present a novel architecture for handling a natural language action space, called a deep reinforcement relevance network (DRRN), and demonstrate its effectiveness in two tasks. We present results showing the new model can generalize to paraphrased actions rather than just memorizing state-action relations in experiments with text games. It is also shown to be useful in experiments on predicting popular Reddit threads with tracking a single discussion thread.

In Chapter 5 we further investigate predicting popular Reddit threads in a combinatorial action space scenario, and address how to better model dependency between sub-actions as well as reducing search complexity in picking the optimal action.

Unlike reinforcement learning for a domain-specific task, reinforcement learning in natural language scenarios often involve understanding about the world knowledge. The architectures in Chapter 4 and 5 model state embeddings from the local context of the environment. In Chapter 6 we propose a natural framework for incorporating external knowledge to form a global context conditioned on current state of the environment. We verify with experiments that this framework improves reinforcement learning performance in natural language scenarios. We also explore temporal context as another form of global context.

Chapter 7 concludes this thesis with a summary and points out some open questions for future work in this area.

Chapter 2
BACKGROUND

Our goal is to study long-term decision making in natural language scenarios, where an agent learns to understand a language environment and pick the right action based on language. This chapter presents a summary of the past and current research related to deep reinforcement learning in natural language scenarios. Deep reinforcement learning is a relatively new research topic (at the time this thesis is written). This thesis combines recent advances in deep learning (Section 2.1), deep learning specifically for natural language processing (Section 2.2), and reinforcement learning (Section 2.3). It builds on recent work in deep reinforcement learning, specifically using deep neural networks as function approximations in a variety of applications (Section 2.4).

2.1 Deep Learning

Artificial intelligence is a thriving field with many practical applications and active research topics. Perhaps the most powerful and ground-breaking development over the past decade is deep learning [34, 91, 48]. Traditionally, many artificial intelligence tasks depend on designing the right set of features to extract for that task. For example, speech scientists studied the human hearing system and proposed Mel-frequency cepstrum coefficients [32] as a representation of the short-term power spectrum of a sound. In computer vision, the scale-invariant feature transform [106] and the histogram of oriented gradients [29] are widely used as a first step in detecting objects in an image.

A common research trend in all artificial intelligence applications is that algorithms are moving towards a more end-to-end trainable direction. This is due to multiple reasons. First, many applications, such as speech recognition or objection detection in images, have much

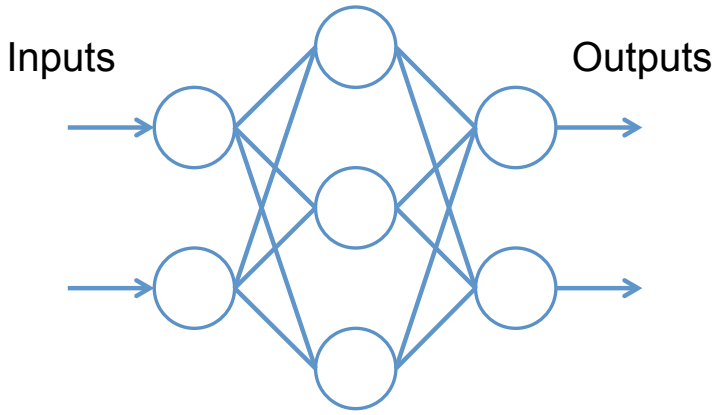


Figure 2.1: A multilayer perceptron with input dimension 2, output dimension 2, and one hidden layer with hidden dimension 3

more data than what was available ten years ago. With a powerful end-to-end model, it is possible to learn a feature representation that is even more expressive than hand-engineered features that were devised over the years. Second, for many tasks it is difficult to know what features should be extracted. Especially when a new task/domain is studied, it takes effort to design features that will work in the new situation. Third, the development of faster parallel computation through hardware such as a graphics processing unit (GPU) enables training algorithms to converge faster.

Deep learning refers to using a deep neural network for function approximation. The simplest type of deep neural network is a feedforward neural network, or sometimes called a multilayer perceptron. The model has a layer-wise structure describing different functions composing from input side to output side. For example, we might have three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ connected in a chain, to form $f(\boldsymbol{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x})))$, where $\boldsymbol{x}$ is an input vector and $f(\boldsymbol{x})$ is an output vector, as illustrated in Figure 2.1.

A common choice of parametric function ($f^{(1)}$, $f^{(2)}$, or $f^{(3)}$) is an affine transformation followed by a nonlinear element-wise activation function. The activation function can be sigmoid [104], tanh, or rectified linear unit [45]. Thus for a particular type of problem (such as regression, classification, or unsupervised clustering), given the objective function (such

as mean squared error, cross entropy, or log likelihood), learning the model is formulated as an optimization process. A nice property with most deep neural networks is that the gradient (first-order derivative) of the objective function with respect to the model parameters, can be computed in closed-form, using back-propagation algorithm [139]. Thus, gradient-based optimization techniques, such as mini-batch gradient descent, are widely used in deep learning.

Convolutional neural networks (CNN) and recurrent neural networks (RNN) are two other types of deep architectures. In a CNN there are usually two types of operations, the convolution operation and the (max) pooling operation. For example, given a two-dimensional image I and a two-dimensional kernel (filter) K, the output of the convolutional layer is:

$$S(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n).$$

After a nonlinear activation function, a pooling function is used to further modify the output. This operation replaces the output of the net at a certain location with a summary statistic (usually maximum) of the nearby neighboring outputs.

An RNN is especially suitable for modeling sequential data. Given an input sequence $\{\boldsymbol{x}^{(t)}\}$, the output units are represented as a function of both current input and previous output units: $\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)})$. A simple RNN uses an affine transformation followed by a nonlinear activation as the function f. To avoid gradient vanishing/exploding [68, 131] and to give the model capacity for controlling when to remember/forget, long short-term memory networks [69] and gated recurrent unit networks [25] are proposed with gating mechanisms.

Another stream of work in recent deep learning research is the attention mechanism [7, 159, 174], where a probability distribution is computed to pay attention to certain parts of a collection of data. It has been shown that the attention mechanism can handle long sequences or a large collection of data, while being quite interpretable.

So far there are many empirical tricks on how to make deep neural network training better. One of the most widely used trick is to use a learning rate schedule, such as rmsprop [170],

momentum [162], or adam [82]. This is especially helpful when the network has recurrent link or ill-shaped objective function. Another stream of tricks aim to efficiently pass the error signals in back-propagation, including initialization [44], batch normalization [74], highway networks [156] or residue networks [65]. Researchers have also investigated methods that prevent overfitting, such as dropout [155].

One of the early successful large-scale applications of deep learning is in automatic speech recognition, particularly in acoustic modeling [67] replacing the Gaussian mixture model for characterizing state distributions with a hidden Markov model [8]. Work with neural networks in acoustic modeling goes back decades, but it recently took off with advances in methods and hardware for learning deep models. Computer vision has been the most active research area for deep learning. Since 2010, ImageNet project runs an annual contest where software programs compete to correctly detect and classify objects and scenes [140]. Since 2012, deep learning has dominated in ImageNet challenge [86]. Deep learning applications in natural language processing will be discussed in the following section.

2.2 Continuous space language processing

In the natural language processing area, learning language representations, especially using a deep neural network, has become very popular. This is again because handcrafting features is time-consuming and domain-dependent. Learned word representations provide a powerful similarity model, and representations are generally divided into two categories, distributional and distributed. Distributional word representations are based on co-occurrence/context, with the belief that linguistic items with similar distributions have similar usage. Distributed representations are dense and continuous in the coordinate of representation space, and often work even better than distributional similarity. The main advantage of having a distributed representation of text is that the text embedding is compact and less susceptible to data sparsity.

Early distributed models were based on linear transforms of distributional statistics. In unsupervised tasks such as language modeling, Bengio et al. [13] first introduced a neu-

ral probabilistic language model that learns a distributed representation of words. Mnih & Hinton [117] proposed three graphical models for statistical language modeling and showed the advantage of using a log-bilinear language model. In speech recognition, Schwenk [148] described using a neural language model interpolated with the back-off language model to achieve consistent word error rate reduction. Mikolov et al. [115] proposed a simple recurrent neural network based language model and provide ample empirical evidences that connectionist language models are superior to standard n-gram techniques. Mikolov et al. [114] introduced word2vec, which is an efficient estimation of continuous vector representations of words. They further explore distributed representations of sentences and documents [90] and show that variable-length pieces of texts can be represented by a dense fixed-length vector. Pennington et al. [133] presented a log-bilinear model that combines the advantages of global matrix factorization and local context window methods. Kiros et al. [83] described an approach for unsupervised learning of reconstructing the surrounding sentences of an encoded passage using an encoder-decoder model.

In supervised (task-oriented) learning, Collobert and Weston [26] described a convolutional neural network architecture that is trained jointly with multitask learning. In web search areas, Huang et al. [73] developed the Deep Structured Semantic Model that uses deep neural network to approximate this projection at semantic level, and measure relevance by computing cosine similarity. Recently, Chen et al. developed a fully discriminative learning approach for supervised Latent Dirichlet Allocation model using mirror-descent back propagation [22] and demonstrated its interpretability on several large-scale text classification tasks [21]. These related work shows variants for language representation in various tasks, and provides insights when designing deep architectures for reinforcement learning.

2.3 Reinforcement learning

Reinforcement learning is the task of learning to make decisions that maximize rewards over a period of time [163]. This section reviews the concepts of traditional reinforcement learning and recent advances of deep reinforcement learning, and their non-text applications such as

TD-gammon [169] and Atari games [121, 119].

In the reinforcement learning setting, an agent interacts with an environment following a particular policy. Each time the agent takes an action, the environment feeds back a numerical value, called a reward, or reinforcement. The goal is to learn an optimal policy that maximizes the expected sum of discounted rewards over a period of time.[1] More formally, we denote the environment state at time t as $s_t \in \mathcal{S}$, where $\mathcal{S}$ is the set of all possible states. The agent's action at time t is a_t, and one time step later, the agent receives a scalar reward r_{t+1}, and the state changes to s_{t+1}. As time goes on, this agent-environment interaction is characterized as:

$$s_0 \xrightarrow[r_1]{a_0} s_1 \xrightarrow[r_2]{a_1} s_2 \rightarrow \ldots \rightarrow s_t \xrightarrow[r_{t+1}]{a_t} s_{t+1} \rightarrow \ldots$$

The above illustration is called an episode, and can also be characterized by a sequence of tuples $(s_t, a_t, r_{t+1}, s_{t+1})$. A policy $\pi(s)$ is a mapping from state to action, or probability $\pi(s, a)$ of taking action a at state s. The goal of reinforcement learning is to learn an optimal policy that brings maximum rewards for any given state.

Markov Decision Process

Typically the transition in a reinforcement learning setting is stochastic, i.e. we can use a probability distribution to characterize the next possible state and reward given the current state and action, as well as previous history information (rewards/states/actions). A reinforcement learning problem that satisfies the Markov property is called a Markov Decision Process (MDP). For an MDP, we have:

$$Pr\left[s_{t+1}, r_{t+1} | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0\right] = Pr\left[s_{t+1}, r_{t+1} | s_t, a_t\right],$$

which greatly reduces the number of model parameters.

An important notion in reinforcement learning is the value function, which is a function of the state (denoted as $V^\pi(s)$), or state-action pair (denoted as $Q^\pi(s, a)$) under a particular

[1] The discount rate determines the present value of future rewards. It decides how myopic or long-sighted the agent will be, and also helps to make the value function converge.

policy π. The value function computes the expected sum of discounted rewards. If we denote the discounting factor as $\gamma \in [0, 1]$, then the value function of state s under a policy π is:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{i=0}^{\infty} \gamma^{i} r_{t+1+i} \mid s_t = s\right],$$

and the value function of state-action pair (s, a) under policy π is:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[\sum_{i=0}^{\infty} \gamma^{i} r_{t+1+i} \mid s_t = s, a_t = a\right].$$

The state value function is used when the value of the environment can be fully characterized using the state representation, such as in chess or game of Go, while the action value function is used if both state and action affect the value of the environment.

The Bellman equation [137] describes the "Principle of Optimality" of the optimal value function, i.e.

> An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.[2]

The Bellman equation is a necessary condition for optimizing the action policy for an MDP. Take the famous Q-learning [179] as an example, if we denote the optimal value function as $Q^{*}(s, a)$, its corresponding Bellman's equation is:

$$Q^{*}(s_t, a_t) = \mathbb{E}_{\pi}\left[r_{t+1} + \gamma \max_{a} Q^{*}(s_{t+1}, a)\right].$$

Inspired by fixed-point iteration and stochastic gradient descent, Q-learning solves the Bellman equation by performing the following update, with the observed tuple $(s_t, a_t, r_{t+1}, s_{t+1})$:

$$Q(s_t, a_t) \leftarrow (1 - \eta) Q(s_t, a_t) + \eta \left(r_{t+1} + \gamma \max_{a} Q(s_{t+1}, a)\right),$$

where η, sometimes denoted as $\eta_t(s_t, a_t)$, is the learning rate.

---

[2] From Bellman, 1957, Chap. III.3.

Partially Observable Markov Decision Process

Often times the agent cannot get full information about the current state. This is called a Partially Observable Markov Decision Process (POMDP). Partial observation of the state can appear in the case of game play when given a current observation a player's history action still affects future rewards. For example, in a text-based game, a player may need to collect enough money in order to purchase certain key items. The game may or may not tell the player how much he needs to accumulate. In the latter case, the game state is partially shown to the player. Another example is from a spoken dialogue system, where a user speaks to a machine in order to complete a particular task, such as restaurant search or buying a flight ticket. The spoken dialogue system needs to first recognize the user's speech, and track the user's intent throughout the entire dialogue. In this case, automatic speech recognition might be erroneous and the user's intent is only partially observable to the dialogue manager.

There are a couple of approaches for transforming a POMDP to an MDP. The first is by defining the "state" to be all history information ((s_i, a_i, r_i), for $i = 0, 1, 2, \ldots$). An obvious drawback is the number of states will increase vastly, and the dimension of the state increases at every time step, making solving the problem even more difficult. Another approach, called belief state, is to maintain a probability distribution over all possible states. In the case of dialogue state tracking, the belief state characterizes the user's intention, often with human-specified features (e.g. size of pizza order). The belief state is updated in a posterior probability fashion after the system processes the current user input.

In our applications, both text games and predicting popular Reddit threads are essentially POMDPs, but we treat them as MDPs for simplicity. One can reduce the effect of the non-observable part by considering longer history of the state. This is exactly what we did with predicting popular Reddit threads. For text games, we did not consider transforming a POMDP to an MDP, because including the history for games could lead to memorizing the game since the game space is small. As discussed further in Chapter 7, it will be interesting future work to compare methods such as DRQN [57] or recurrent reinforcement learning [97]

TD-method	Value function	On/off-policy	Bellman equation
TD(0)	$V(s)$	on-policy	$V(s_t) = \mathbb{E}_\pi \left[r_{t+1} + \gamma V(s_{t+1}) \right]$
SARSA	$Q(s, a)$	on-policy	$Q(s_t, a_t) = \mathbb{E}_\pi \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) \right]$
Q-learning	$Q(s, a)$	off-policy	$Q(s_t, a_t) = \mathbb{E}_\pi \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) \right]$
Actor-Critic	actor: $p(a\|s)$ critic: $V(s)$	on-policy	

Table 2.1: Comparison of four temporal difference methods

as a solution to model a POMDP.

Temporal-Difference Learning

Temporal-difference learning is one of the three fundamental classes of algorithms solving reinforcement learning; the other two are dynamic programming and Monte Carlo methods [163]. The idea of temporal-difference learning is to bootstrap trained models, i.e. to update models partly based on model predictions, without waiting for final outcomes. Compared to the other two methods, dynamic programming requires the knowledge of the environment, and Monte Carlo cannot do a step-by-step incrementation thus it lacks the advantages of bootstrapping. TD(0), SARSA, Q-learning, and Actor-Critic are all different variations of temporal difference learning. They differ in the choice of value functions, on/off-policy (on-policy means policy evaluation is done according to data policy, or behavior policy), Bellman equations etc., as shown in Table 2.1. These differences result in different update rules, as shown in Table 2.2. In this thesis, we will stick with Q-learning, because empirically it gives better convergence behavior, and the tasks studied are essentially evaluating values in given natural language actions.

The tradeoff between exploration and exploitation is an important issue broadly and often involves designing sophisticated action selection algorithms. Based on a specific policy,

TD-method	Update rule
TD(0)	$V(s_t) \leftarrow (1-\eta)V(s_t) + \eta(r_{t+1} + \gamma V(s_{t+1}))$
SARSA	$Q(s_t, a_t) \leftarrow (1-\eta)Q(s_t, a_t) + \eta(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}))$
Q-learning	$Q(s_t, a_t) \leftarrow (1-\eta)Q(s_t, a_t) + \eta(r_{t+1} + \gamma \max_a Q(s_{t+1}, a))$
Actor-Critic	$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$; $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta\delta_t$

Table 2.2: Update rules for four temporal difference methods

selecting the action with the highest value function is called greedy action selection, and is preferred if we want to maximize exploitation. However, in the long run, especially in a dynamic environment, it is often preferable to keep exploring new strategies as well. ϵ-greedy and softmax action selection are two popular algorithms. In ϵ-greedy, actions are picked at random with a small probability ϵ, and greedy action with probability $1-\epsilon$. That is:

$$a_t = \begin{cases} \arg\max_{a_t^i \in \mathcal{A}_t} Q(s_t, a_t^i)), \text{ with } p = 1-\epsilon \\ \text{uniformly random } a_t^i \in \mathcal{A}_t, \text{ with } p = \epsilon \end{cases}$$

In softmax action selection, a softmax function is applied for value function $Q(s,a)$ to form a discrete probability distribution, and the action is selected based on this multinomial distribution. More specifically, the action is chosen according to the following probability:

$$\pi(a_t = a_t^i | s_t) = \frac{\exp(\alpha \cdot Q(s_t, a_t^i))}{\sum_{j=1}^{|\mathcal{A}_t|} \exp(\alpha \cdot Q(s_t, a_t^j))},$$

where $\mathcal{A}_t$ is the set of feasible actions at state s_t, a_t^i is the i-th feasible action in $\mathcal{A}_t$, $|\cdot|$ denotes the cardinality of the set, and α is the scaling factor in the softmax operation. In this thesis, the choice of whether ϵ-greedy or softmax action selection is empirical and based on convergence speed in learning curves.

Another stream of work focuses on policy gradients [164, 9, 5, 185]. In policy gradient methods, a differentiable policy with respect to its weights is modeled directly. The policy gradient theorem states that the gradient of expected reward with respect to the policy

parameters, can be written in a form suitable for estimation from experience aided by an approximate action-value or advantage function [164]. Examples of this approach include the popular REINFORCE method, which appears in several recent NLP papers [93, 187, 92], the actor-critic method, which is used in human-computer dialog systems [157], and asynchronous methods for Atari games [120] and continuous control [99]. Amari [5] proposed a natural policy gradient using the Fisher-information matrix to estimate policy gradients when the parameter space has a certain underlying structure (Riemannian structure).

Function approximation is widely used in reinforcement learning, in both value function approaches and policy gradient methods. The motivation is that it is often unnecessary or impossible to represent value functions or policies using a lookup table. There has been some well-developed theory guaranteeing convergence properties in using a lookup table or linear function approximation in temporal difference learning. However, little has been proved when function approximations are non-linear and the update rules (Table 2.2) follow direct stochastic gradient descent. Baird [9] developed the residual algorithm, which is a generalization of direct and residual gradient algorithms. It has guaranteed convergence compared to direct algorithms and it converges faster compared to residual gradient algorithms. In this thesis, the focus is on evaluating action values in natural language scenarios, so we use Q-learning rather than policy gradient methods, and we do not address theoretical aspects (such as bounds or convergence properties) of proposed algorithms.

2.4 Deep reinforcement learning

Most successful reinforcement learning applicability has been limited to domains where critical hand-crafted features are available. Recently, inspired by advances in deep learning [91, 67, 86, 28], significant progress has been made by combining deep learning with reinforcement learning, thus called deep reinforcement learning (DRL). However, for reinforcement learning, approximating the value function using a non-linear transformation is known to be unstable or even diverge [172]. The "Deep Q-Network" (DQN) was developed and applied to Atari games (Figure 2.2(a)) and was shown to achieve human level performance by applying

convolutional neural networks to the raw image pixels [118, 119]. A deep neural network is used as a function approximation in a variant of Q-learning [179], and techniques such as target network and experience replay, are introduced to ensure the algorithm converges stably.

In a more recent development, Schaul et al. [145], study two variants of prioritized sampling (rank-based and proportional) as substitutes for uniformly sampling experience transitions. Knowledge transfer and multi-task/transfer learning are pursued by Andrei et al. [141] and Parisotto et al. [130]. Wang et al. [178] proposes a new model called dueling network architectures that maintaining separate value and (action) advantage functions, and show that this architecture leads to better policy evaluation in the presence of many similar-valued actions. Their evidence that the state-value function and (action) advantage function may have very different value ranges suggests it is better to learn state and action sides separately.

To address overestimations of action values, double Q learning [55, 173] has been proposed as a new off-policy algorithm, and it leads to better performance gains on several Atari games. Dulac-Arnold et al. [38] also investigated a problem of large discrete action spaces. A Wolpertinger architecture is proposed to reduce computational complexity of evaluating all actions. While a combinatorial action space can be large and discrete, this method does not directly apply in our case, because the possible actions are changing over different states. In this thesis, we borrow the philosophy from double Q and propose a two-stage Q-learning approach to reduce computational complexity in a combinatorial natural language action space.

Other work that targets a structured action space includes: an actor-critic algorithm, where actions can have real-valued parameters [56]; and the factored Markov Decision Process (MDP) [50, 142], with certain independence assumptions between a next-state component and a sub-action. As for a bandits setting, Yue and Guestrin [193] considered diversification of multi-item recommendation, but their methodology is limited to using linear approximation with hand-crafted features.

2.4.1 DRL applications

TD-gammon is one of the early successful applications using temporal-difference learning and neural network function approximations [169]. It was able to achieve the expertise level of a human player, and its extensive exploration in training led to advances in backgammon strategy. Atari games (Figure 2.2(a)) are another example of applying deep reinforcement learning [121, 119]. The algorithm is capable of human level performance by reading in raw image pixels and applying convolutional neural networks. Experience replay and iterative updates are used to make sure the non-linear function approximation (such as a deep neural network) converges stably. It is demonstrated that a single architecture can successfully learn control policies in a range of different environments with little prior knowledge about game design.

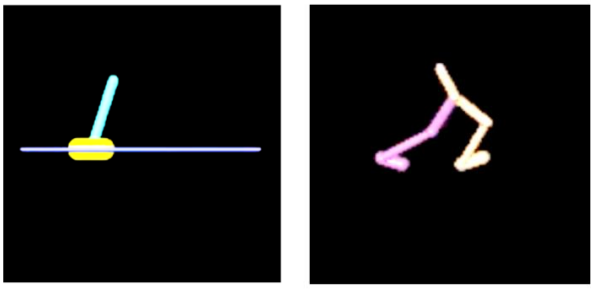
Another stream of work focuses on continuous control (Figure 2.2(b)) with deep reinforcement learning [99], where an actor-critic algorithm operates over a known, continuous action space. Tasks described in this thesis have an action space defined through unrestricted natural language. That is: i) inherently discrete, but mapped to a continuous space for more effective value function approximation, and ii) unbounded in that sequences can be any length.

There has also been increasing interest in applying reinforcement learning, especially DQN, to other problems. Li et al. [97] developed a joint training approach for recurrent reinforcement learning and demonstrate its effectiveness on a customer relationship management task. Silver et al. [150] combined deep neural networks and Monte Carlo tree search and developed a computer program that became the first Computer Go program to beat a human professional Go player on a full-sized 19×19 board (Figure 2.2(c)).

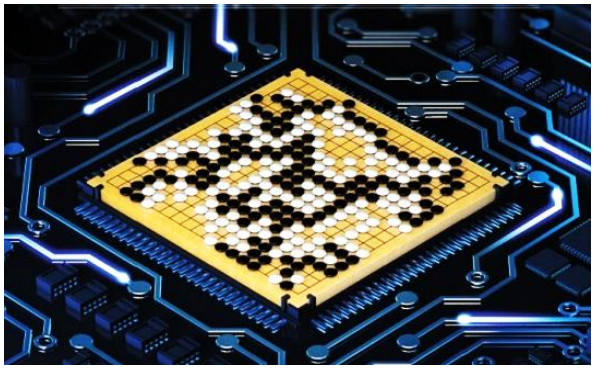
In terms of testbeds, the Arcade Learning Environment (ALE) [12] is a framework composed of Atari 2600 games to develop and evaluate AI agents. So far the largest and the most comprehensive testbeds for reinforcement learning are from OpenAI. OpenAI Gym[3] is

[3]https://gym.openai.com

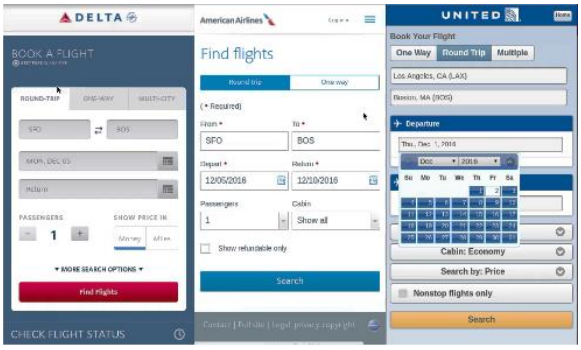
(a) Atari game “Space Invader” (from [121])



(b) Continuous control (from [99])



(c) AlphaGo (from [150])



(d) OpenAI Universe platform: interacting with a mobile user interface

Figure 2.2: DRL applications

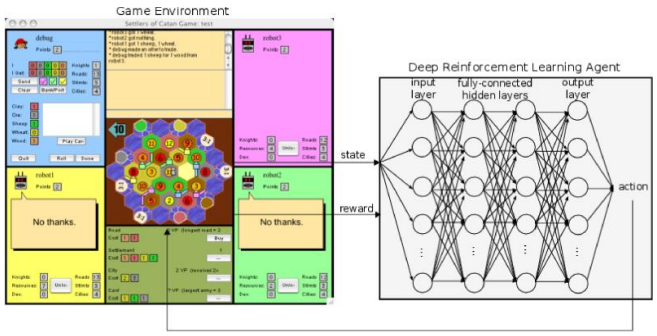
a toolkit for the developing and comparing reproducible reinforcement learning algorithms. Their environment consists of classic control, board games, to video games such as Atari games and Doom. OpenAI Universe[4] takes one step further, providing a software platform for evaluating and training intelligent agents across the world’s supply of games, websites and other applications. Universe has already integrated many environments, including flash games, browser tasks like Mini World of Bits, and real-world browser tasks (Figure 2.2(d)). Recently, Grand Theft Auto V was added to Universe for self-driving vehicle simulation.

[4]https://universe.openai.com

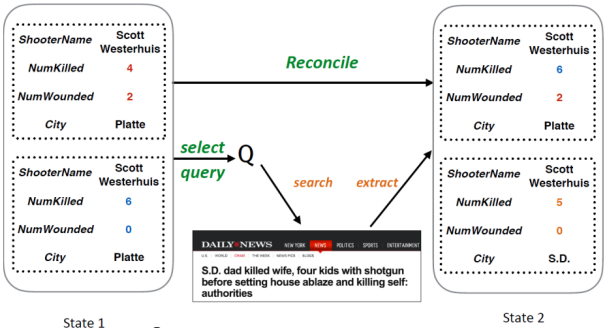
2.4.2 DRL applications in language processing

In language processing, reinforcement learning has been applied to a dialogue management system that converses with a human user by taking actions that generate natural language [146, 151]. Very recently, Hongyu Guo [51] introduced a novel schema for sequence-to-sequence learning with deep reinforcecement learning, applied on a text generation task. Cuayáhuitl et al. [27] describes a successful situated dialogue setting (Figure 2.3(a)) with deep reinforcement learning for training intelligent agents with strategic conversational skills. There has also been interest in extracting textual knowledge to improve game control performance [19], and mapping natural language instructions to sequences of executable actions [18]. Narasimhan et al. [122] applied a Long Short-Term Memory DQN framework to the task of learning control policies for parser-based text games, which achieves higher average reward than the random and Bag-of-Words DQN baselines. Due to the potentially infinite input space, modeling parser-based text games requires restrictions on player input [122], such as fixed command structures (one action and one argument object), and limited action-side vocabulary size. For unrestricted natural language, this approach is infeasible.

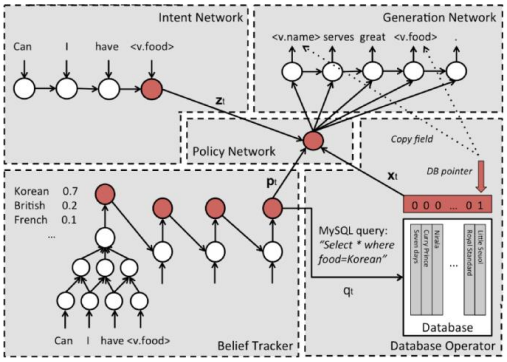
Narasimhan et al. [123] applied reinforcement learning for acquiring and incorporating external evidence to improve information extraction accuracy (Figure 2.3(b)). They use Q-learning but does not have a natural language action space. Wen et al. [180] introduced a neural network-based trainable dialogue system along with a way of collecting task-oriented dialogue data based on a pipelined Wizard-of-Oz framework (Figure 2.3(c)). Su et al. [157] described a unified neural network framework to first learn by supervision from a set of dialogue data and then continuously improve its behaviour via reinforcement learning, all using gradient-based algorithms on one single model. Li et al. [93] applied deep reinforcement learning to model future reward in conversation-style chatbot dialogue. Their model simulates dialogues between two virtual agents, using policy gradient methods to reward sequences that display several specific useful conversational properties (Figure 2.3(d)). Bhuwan et al. [35] proposed a dialogue agent that provides users with an entity from a knowledge base by



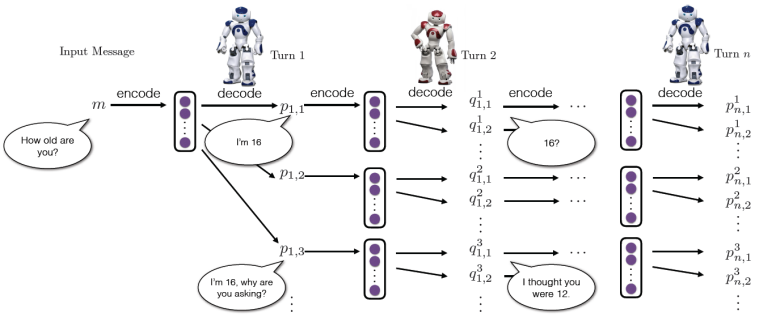
(a) Board game using language (from [27])



(b) DRL for improving information extraction accuracy (from [123])



(c) Task-oriented dialog (from [180])



(d) Chatbot (from [93])

Figure 2.3: DRL applications in language processing

interactively asking for its attributes. These efforts aim at *generating* natural language responses as actions using policy gradient methods, and do not directly apply in our scenarios involving text games or predicting popular Reddit threads, where the possible actions are provided. In terms of testbeds, Nogueira and Cho [125] have proposed a goal-driven web navigation task for language based sequential decision making study. However their paper only uses supervised learning rather than reinforcement learning for benchmark performance.

This thesis introduces a new class of text games and an online popularity prediction and tracking task as a benchmark task for reinforcement learning with a combinatorial, natural language action space.

2.5 Limitations of prior work

This chapter reviews necessary background related to deep learning in natural language processing and deep reinforcement learning. These studies can be roughly divided in two categories. One is using value function approaches such as Q-learning for dealing with a natural language state space and a pre-specified action space, as in the cases of board games, information extraction, or parser-based text games. The other is using policy gradient methods for generating a natural language response (such as task-oriented dialog and chatbot), and often post-date our initial work in Chapter 4. In our scenarios, both states and actions are given as unrestricted natural language text, and the agent tries to evaluate the future value in each action, so the above techniques cannot be directly applied. In the following chapters, we will first describe our approach to a natural language action space, and then further extend the architecture and investigate a combinatorial action space that is common in a recommendation system such as predicting and tracking popular discussion threads. Also, one distinctive aspect of natural language tasks is that incorporating external knowledge is usually helpful. This thesis serves to extend deep reinforcement learning methods to be more effective in a variety of natural language scenarios.

Chapter 3

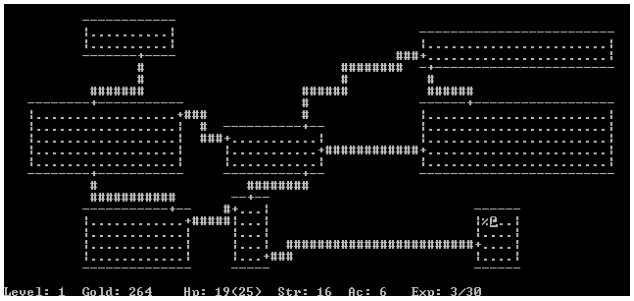
TASKS

3.1 Text Games

One of the early text game subgenres is roguelike games (Figure 3.1(a)). The game is turn-based and often with tile-based graphics design. At the start of the game, the main character is placed at the top level of a dungeon, with basic equipment such as a simple weapon, armor, torches, and food. The player moves the character through the dungeon rooms, collecting treasure or fight enemies. The map and main character are shown as ascii characters, and some information about the environment and main character is shown in text form. However, in this kind of game, the state information is conveyed through graphics using text characters rather than natural language, and navigating through the map requires no language understanding. Thus we do not focus on this type of game in this thesis.

There are three other types of text games: parser-based (Figure 3.1(b)), choice-based (Figure 3.1(c)), and hypertext-based (Figure 3.1(d)). Parser-based games (e.g. Zork[1]) were popular among early personal computer users. They are the least user-friendly text games. Their prominent feature involves a natural-language parser inside the simulator that accepts typed-in commands from the player, usually in the form of verb phrases, such as "eat apple", "get key", or "go east." Choice-based and hypertext-based games, on the other hand, present actions described in text after or embedded within the state text. The player chooses one of these actions, and the story continues based on the action taken at this particular state. With the development of web browsing and richer HTML display, choice-based and hypertext-based text games have become increasingly popular in online communities, increasing in

[1]https://en.wikipedia.org/wiki/Zork



(a) Roguelike text game

Front Steps

Well, here we are, back home again. The battered front door leads north into the lobby.

The cat is out here with you, parked directly in front of the door and looking up at you expectantly.

>_

(b) Parser-based

Well, here we are, back home again. The battered front door leads into the lobby.

The cat is out here with you, parked directly in front of the door and looking up at you expectantly.

- **Step purposefully over the cat and into the lobby**
- **Return the cat's stare**
- **"Howdy, Mittens."**

(c) Choiced-based

Well, here we are, back **home** again. The **battered front door** leads into the lobby.

The cat is out here with you, parked directly in front of the door and **looking up at you expectantly**.

You're **hungry**.

(d) Hypertext-based

Figure 3.1: Different types of text games

percentage of online text games from 8% in 2010 to 62% in 2014,[2] as shown in Table 3.1.[3]

Year	2010	2011	2012	2013	2014
Percentage	7.69%	7.89%	25.00%	55.56%	61.90%

Table 3.1: Percentage of choice-based and hypertext-based text games since 2010, in archive of interactive fictions

For a specific parser-based text game, Narasimhan et al. [122] have defined a fixed set of 222 actions, which is the total number of phrases the parser accepts. Thus the parser-based text game is reduced to a problem that is well suited to a fixed-action-set DQN.

[2]Statistics are obtained from http://www.ifarchive.org

[3]Statistics are obtained from http://www.ifarchive.org

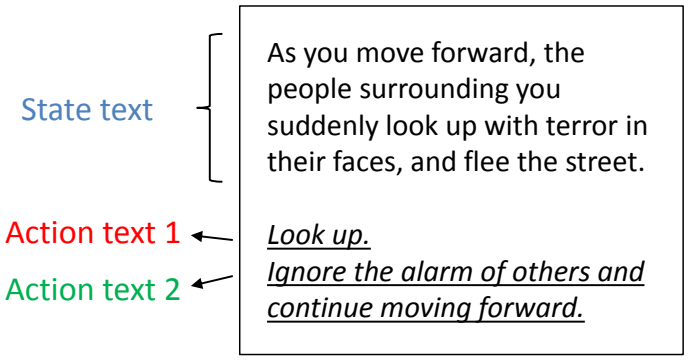


Figure 3.2: State and actions in a text game

However, for choice-based and hypertext-based text games, the size of the action space could be exponential with the length of the action sentences, which is handled here by using a continuous representation of the action space. In Table 3.2, we show basic game statistics of the two popular text games used in this thesis: "Saving John" and "Machine of Death". Simulators are available at `https://github.com/jvking/text-games`.

In Figure 3.1, we show one time step in the game "Machine of Death". In this case, the state text (describing the state of environment) is s_t = "As you move forward, the people surrounding you suddenly look up with terror in their faces, and flee the street." We see two action texts $\mathcal{A}_t = \{a_t^1, a_t^2\}$ and their underlines indicate that they have hyperlinks connecting to possible next states. In this time step, a_t^1 = "Look up." and a_t^2 = "Ignore the alarm of others and continue moving forward." The agent needs to predict $Q(s_t, a_t^1)$ and $Q(s_t, a_t^2)$ in order to make a decision.

In terms of reward, we manually annotate terminal rewards for all distinct endings in both games. The magnitude of reward scores are given to describe sentiment polarity of good/bad endings, as shown in Table 3.3 and Table 3.4. For each non-terminating step we assign with a small negative reward, to encourage the learner to finish the game as soon as possible.

Game	Saving John	Machine of Death
Text game type	Choice	Choice & Hypertext
State vocab size	1762	2258
Action vocab size	171	419
Avg. words/description for the state	76.67	67.80
State transitions	Deterministic	Stochastic
# of states (underlying)	≥ 70	≥ 200

Table 3.2: Statistics for the games "Saving John" and and "Machine of Death".

Reward	Endings (partially shown)
-20	Suspicion fills my heart and I scream. Is she trying to kill me? I don't trust her one bit...
-10	Submerged under water once more, I lose all focus...
0	Even now, she's there for me. And I have done nothing for her...
10	Honest to God, I don't know what I see in her. Looking around, the situation's not so bad...
20	Suddenly I can see the sky... I focus on the most important thing - that I'm happy to be alive.

Table 3.3: Final rewards defined for the text game "Saving John"

3.2 Predicting Popular Reddit Threads

Social media such as online discussion forums have become a popular way for people to express their opinions and follow discussions that interest them. Discussion forums with a lot of web text, such as reddit, have also drawn interests in natural language processing community [76, 182, 4]. This is because web texts are quick reflections on news issues and society trends [87], and can be useful in sentiment analysis and opinion mining [126, 113]. They are also more challenging since they are more flexible than formal text, e.g. news

articles. Our experiments are based on Reddit,[4] one of the world's largest public discussion forums. Reddit is a social networking and news website founded in 2005, and has 1.7 billion comments publicly available for research.[5]

On Reddit, registered users initiate a post and people respond with comments, either to the original post or one of its associated comments. Together, the comments and the original post form a discussion tree, which grows as new comments are contributed. It has been shown that discussions tend to have a hierarchical topic structure [182], i.e. different branches of the discussion reflect a narrowing of higher level topics. Reddit discussions are grouped into different domains, called subreddits, according to different topics or themes. Depending on the popularity of the subreddit, a post can receive hundreds of comments.

Popularity prediction and tracking in the Reddit setting is used in this thesis for studying reinforcement learning to model long-term rewards in a combinatorial action space. At each time step, the state corresponds to the collection of comments previously recommended. The system aims at automatically picking K lines of the discussion to follow from the new set of N comments in a given window, which is a combinatorial action.

Comments (and posts) are associated with positive and negative votes (i.e., likes and dislikes) from registered users that are combined to get a *karma score*, which is used as the measure for popularity. An example of the top of a Reddit discussion tree is given in Figure 3.3. The scores in red boxes mark the current karma (popularity) of each comment, and it is quite common that a lower karma comment (e.g. "Yeah, politics aside, this one looks much cooler", compared to "looks more like zom-bama") will lead to more children and popular comments in the future (e.g. "true dat"). In Figure 3.4, we show another example where a user's comment ("Does the UK have minimums for officials ages?") sparked a discussion and downstream comments have higher karma scores than the first one. Note that the karma scores are dynamic, changing as readers react to the evolving discussion and eventually settling down as the discussion trails off. In a real-time comment recommendation system,

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

[5] https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment

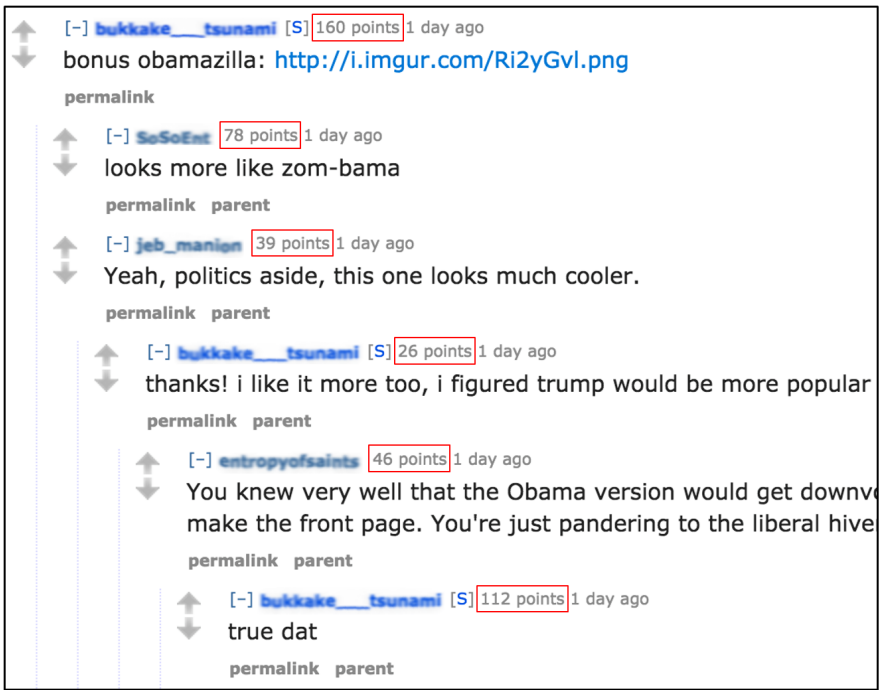


Figure 3.3: A snapshot of the top of a Reddit discussion tree, where karma scores are shown in red boxes.

the eventual karma of a comment is not immediately available, so prediction of popularity is based on the text in the comment in the context of prior comments in the subtree and other comments in the current time window.

To formulate this problem into Q-learning, suppose the agent is at a time step where the set of comments that are being tracked is the top level comment in Figure 3.3 (i.e. ["bonus obamazilla: http://i.imgur.com/Ri2yGvl.png"]). All previously tracked comments are considered as state s_t = ["bonus obamazilla: http://i.imgur.com/Ri2yGvl.png"]. If we set the window size $N = 2$, the agent might only see two actions $\mathcal{A}_t = \{a_t^1, a_t^2\}$, and a_t^1 = "looks more like zom-bama", a_t^2 = "Yeah, politics aside, this one looks much cooler." A good agent will predict $Q(s_t, a_t^1) < Q(s_t, a_t^2)$ and choose the second action that leads to more future karma scores. The formulation of this example is shown in Table 3.5.

In this work, we only consider new comments associated with the threads of the discussion

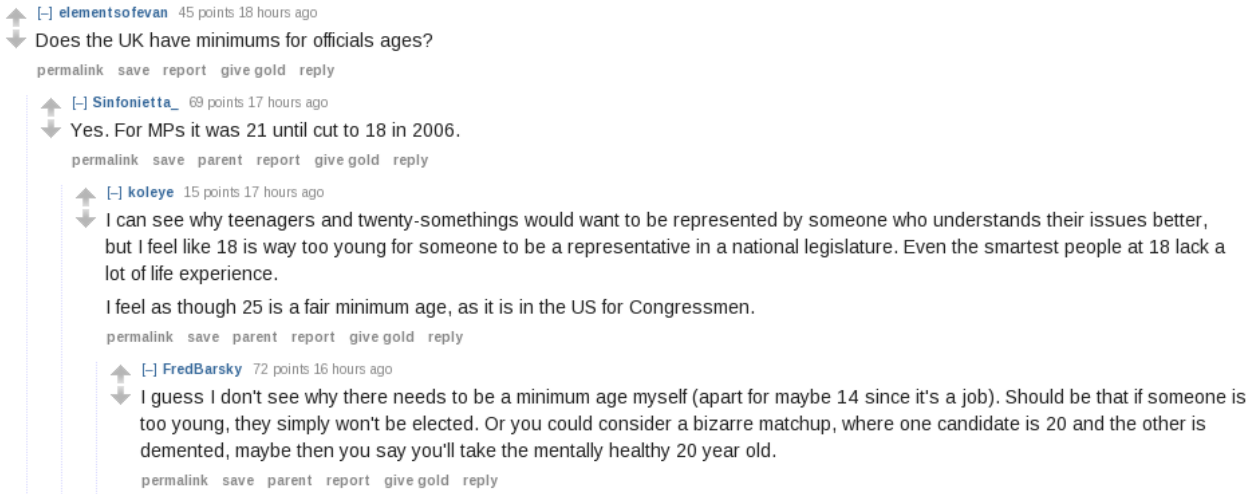


Figure 3.4: Another example showing a person's comment sparked a discussion

that we are currently following to limit the number of possible sub-actions at each time step and with the assumption that prior context is needed to interpret the comments. In other words, the new recommendation should focus on comments that are in the subtrees of previously recommended comments. Typically, one would expect some interdependencies between comments made in the same window if they fall under the same subtree, because they correspond to a reply to the same parent. In addition, there may be some temporal dependency, since one sub-action may be a comment on the other. These dependencies will affect the combined utility of the sub-actions.

More specifically, we can consider the task of recommending multiple discussion threads at the same time. The set of comments that are being tracked at time step t is denoted as M_t. All previously tracked comments, as well as the post (root node of the tree), are considered as state s_t ($s_t = \{M_0, M_1, \cdots, M_t\}$), and we initialize $s_0 = M_0$ to be the post. An action is taken when a total of N new comments $\{c_{t,1}, c_{t,2}, \cdots, c_{t,N}\}$ appear as nodes in the subtree(s) of M_t, where the comments $c_{t,j}$ are the first N in time that are descendants of one of $c^i_{t-1} \in a_{t-1}$. The agent picks a set of K comments to be tracked in the next time

step $t+1$. Thus we have:

$$a_t = \{c_t^1, c_t^2, \cdots, c_t^K\}, \quad c_t^i \in \{c_{t,1}, c_{t,2}, \cdots, c_{t,N}\} \text{ and } c_t^i \neq c_t^j \text{ if } i \neq j \tag{3.1}$$

and $M_{t+1} = a_t$. At the same time, by taking action a_t at state s_t, the reward r_{t+1} is the accumulated karma scores, i.e. the sum over all comments in M_{t+1}. Note that the reward signal is used in online training, while at model deployment (testing stage), the scores are only used as an evaluation metric.

Following the reinforcement learning tradition, we call tracking of a single discussion tree from start (root node post) to end (no more new comments appear) an *episode*. We also randomly partition all discussion trees into separate training and testing sets, so that texts seen by the agent in training and testing are from the same domain but different discussions. For each episode, depending on whether training/testing, the simulator randomly picks a discussion tree, and presents the agent with the current state and N new comments.

We conduct experiments on data from several subreddits, which cover diverse genres and topics. The URLs are shown in Table 3.6. In order to have long enough discussion threads, we filter out discussion trees with fewer than 100 comments. For each of the subreddits, we randomly partition 90% of the data for online training, and 10% of the data for testing. Models are trained separately for each subreddit and the subreddit vocabulary is the most frequent 5000 words. Our evaluation metric is accumulated karma scores. Simulators are available at `https://github.com/jvking/reddit-RL-simulator`. The basic subreddit statistics are shown in Table 3.7.

Our task of tracking popular Reddit comments is somewhat related to an approach to multi-document summarization described in [31]. A difference with respect to our problem is that the space of text for selection evolves over time. In addition, in our case, the agent does not have access to the optimal policy, in contrast to the SEARN algorithm used in that work.

Reward	Endings (partially shown)
-20	You spend your last few moments on Earth lying there, shot through the heart, by the image of Jon Bon Jovi.
-20	you hear Bon Jovi say as the world fades around you.
-20	As the screams you hear around you slowly fade and your vision begins to blur, you look at the words which ended your life.
-10	You may be locked away for some time.
-10	Eventually you're escorted into the back of a police car as Rachel looks on in horror.
-10	Fate can wait.
-10	Sadly, you're so distracted with looking up the number that you don't notice the large truck speeding down the street.
-10	All these hiccups lead to one grand disaster.
10	Stay the hell away from me! She blurts as she disappears into the crowd emerging from the bar.
20	You can't help but smile.
20	Hope you have a good life.
20	Congratulations!
20	Rachel waves goodbye as you begin the long drive home. After a few minutes, you turn the radio on to break the silence.
30	After all, it's your life. It's now or never. You ain't gonna live forever. You just want to live while you're alive.

Table 3.4: Final rewards for the text game "Machine of Death." Scores are assigned according to whether the character survives, how the friendship develops, and whether he overcomes his fear.

State text
bonus obamazilla: http://i.imgur.com/Ri2yGvl.png
Action texts
[1] looks more like zom-bama [2] Yeah, politics aside, this one looks much cooler.
Immediate rewards
[1] 78 [2] 39

Table 3.5: An example state and actions

Subreddit	**URL**
askscience	https://www.reddit.com/r/askscience/
askmen	https://www.reddit.com/r/askmen/
todayilearned	https://www.reddit.com/r/todayilearned/
askwomen	https://www.reddit.com/r/askwomen/
politics	https://www.reddit.com/r/politics/
worldnews	https://www.reddit.com/r/worldnews/
nfl	https://www.reddit.com/r/nfl/

Table 3.6: URLs of subreddit data sets

Subreddit	**# Posts (in k)**	**# Comments (in M)**	**OOV rates**
askscience	0.94	0.32	14.0%
askmen	4.45	1.06	6.7%
todayilearned	9.44	5.11	13.1%
askwomen	3.57	0.81	7.4%
politics	4.86	2.18	9.7%
worldnews	9.88	5.99	10.8%
nfl	11.73	6.12	15.7%
Total	44.87	21.59	N/A

Table 3.7: Basic statistics of filtered subreddit data sets

Chapter 4

DRL WITH A NATURAL LANGUAGE ACTION SPACE

This chapter introduces a novel architecture for deep RL for NLP, termed the Deep Reinforcement Relevance Network (DRRN). In experiments on both text games and predicting popular Redidt threads, we show that the DRRN provides an effective solution to this problem. Previous deep reinforcement learning studies mainly focus on dealing with a large state space (e.g. a natural language state space), while the action space is pre-specified. When actions are described through natural language, the action space is unbounded. Most of contributions in this chapter are published in [58].

4.1 Deep Reinforcement Relevance Network

In this sequential decision making problem, at each time step t, the agent receives a text string that describes the state $s_t \in \mathcal{S}$ (i.e., "state-text") and picks a text string that describes the action $a_t \in \mathcal{A}$ (i.e., "action-text"), where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively. Here, we assume a_t is chosen from a set of given candidates. In our case both $\mathcal{S}$ and $\mathcal{A}$ are described by natural language. Given the state-text and action-texts, the agent aims to select the best action in order to maximize its long-term reward. Then the environment state is updated $s_{t+1} = s'$ according to a probability $p(s'|s, a)$, and the agent receives a reward r_{t+1} for that particular transition. We define the action-value function (i.e. the Q-function) $Q(s, a)$ as the expected return starting from s and taking the action a:

$$Q(s,a) = \mathbb{E}\left\{\sum_{l=0}^{+\infty} \gamma^l r_{t+1+l} | s_t = s, a_t = a\right\}$$

where $\gamma \in (0, 1)$ denotes a discount factor. The Q-function associated with an optimal policy can be found by the Q-learning algorithm [179]:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta_t \cdot \big(r_{t+1} + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\big)$$

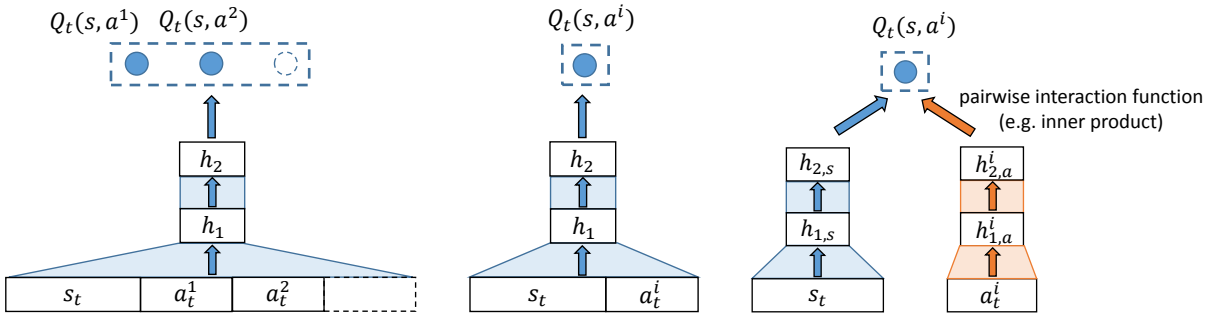
where η_t is a learning rate parameter.

A vanilla Q-learning recursion needs to maintain a table of size $|\mathcal{S}| \times |\mathcal{A}|$, which is problematic for a large state/action space. Prior work using a DNN in Q-function approximation has shown high capacity and scalability for handling a large state space, but most studies have used a network that generates $|\mathcal{A}|$ outputs, each of which represents the value of $Q(s, a)$ for a particular action a. It is not practical to have a DQN architecture of a size that is explicitly dependence on the large number of natural language actions. Further, in many text games, the feasible action set $\mathcal{A}_t$ at each time t is an unknown subset of the unbounded action space $\mathcal{A}$ that varies over time.

For the case where the maximum number of possible actions at any point in time ($\max_t |\mathcal{A}_t|$) is known, the DQN can be modified to simply use that number of outputs ("Max-action DQN"), as illustrated in Figure 4.1(a), where the state and action vectors are concatenated (i.e., as an extended state vector) as its input. The network computes the Q-function values for the actions in the current feasible set as its outputs. For a complex game, $\max_t |\mathcal{A}_t|$ may be difficult to obtain, because $\mathcal{A}_t$ is usually unknown beforehand. Nevertheless, we will use this modified DQN as a baseline.

An alternative approach is to use a function approximation using a neural network that takes a state-action pair as input, and outputs a single Q-value for each possible action ("Per-action DQN" in Figure 4.1(b)). This architecture easily handles a varying number of actions and represents a second baseline.

We propose an alternative architecture for handling a natural language action space in sequential text understanding: the deep reinforcement relevance network (DRRN). As shown in Figure 4.1(c), the DRRN consists of a pair of DNNs, one for the state text embedding and the other for action text embeddings, which are combined using a pairwise interaction



(a) Max-action DQN (b) Per-action DQN (c) DRRN

Figure 4.1: Different deep Q-learning architectures: Max-action DQN and Per-action DQN both treat input text as concantenated vectors and compute output Q-values with a single NN. DRRN models text embeddings from state/action sides separately, and use an interaction function to compute Q-values.

function. The texts used to describe states and actions could be very different in nature, e.g., a state text could be long, containing sentences with complex linguistic structure, whereas an action text could be very concise or just a verb phrase. Therefore, it is desirable to use two networks with different structures to handle state/action texts, respectively. As we will see in the experimental sections, by using two separate deep neural networks for state and action sides, we obtain much better results.

4.1.1 DRRN architecture: Forward activation

Given any state/action text pair (s_t, a_t^i), the DRRN estimates the Q-function $Q(s_t, a_t^i)$ in two steps. First, map both s_t and a_t^i to their embedding vectors using the corresponding DNNs, respectively. Second, approximate $Q(s_t, a_t^i)$ using an interaction function such as the inner product of the embedding vectors. Then, given a particular state s_t, we can select the optimal action a_t among the set of actions via

$$a_t = \arg \max_{a_t^i \in \mathcal{A}_t} Q(s_t, a_t^i). \tag{4.1}$$

More formally, let $h_{l,s}$ and $h_{l,a}$ denote the l-th hidden layer for state and action side neural

networks, respectively. For the state side, $W_{l,s}$ and $b_{l,s}$ denote the linear transformation weight matrix and bias vector between the $(l-1)$-th and l-th hidden layers. $W_{l,a}$ and $b_{l,a}$ denote the equivalent parameters for the action side. In this study, the DRRN has L hidden layers on each side. Note we are abusing notation somewhat here. In the description of the tasks previously, s_t and a_t have been text strings. In the equations describing the DRRN, they correspond to the vector representation of the text string, which in our experiments is a bag-of-words model. A bag-of-words representation computes raw counts of each word in vocabulary and computes a weighted average of word embeddings disregarding word orders.

$$h_{1,s} = f(W_{1,s}s_t + b_{1,s}) \tag{4.2}$$

$$h^i_{1,a} = f(W_{1,a}a^i_t + b_{1,a}) \tag{4.3}$$

$$h_{l,s} = f(W_{l-1,s}h_{l-1,s} + b_{l-1,s}) \tag{4.4}$$

$$h^i_{l,a} = f(W_{l-1,a}h^i_{l-1,a} + b_{l-1,a}) \tag{4.5}$$

where $f(\cdot)$ is the nonlinear activation function at the hidden layers, which, for example, could be chosen as $\tanh(x)$, and $i = 1, 2, 3, ..., |\mathcal{A}_t|$ is the action index. A general interaction function $g(\cdot)$ is used to approximate the Q-function values, $Q(s, a)$, in the following parametric form:

$$Q(s, a^i; \Theta) = g\left(h_{L,s},\ h^i_{L,a}\right) \tag{4.6}$$

where Θ denotes all the model parameters. The interaction function could be an inner product, a bilinear operation, or a nonlinear function such as a deep neural network. In our experiments, the inner product and bilinear operation gave similar results, but the nonlinear neural network using the concatenated state and action space embeddings degraded performance. For simplicity, we present our experiments mostly using the inner product interaction function.

The success of the DRRN in handling a natural language action space $\mathcal{A}$ lies in the fact that the state-text and the action-texts are mapped into separate finite-dimensional embedding spaces. The end-to-end learning process (discussed next) makes the embedding

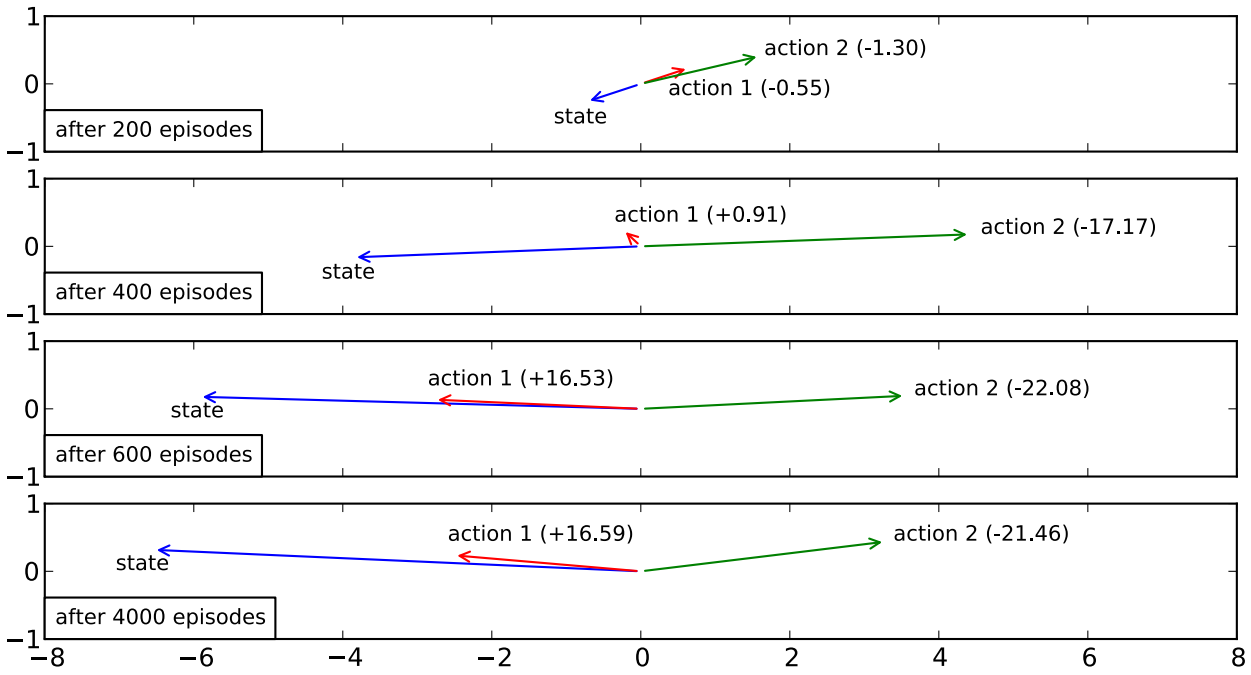


Figure 4.2: PCA projections of text embedding vectors for state and associated action vectors after 200, 400 and 600 training episodes. The state is "As you move forward, the people surrounding you suddenly look up with terror in their faces, and flee the street." Action 1 (good choice) is "Look up", and action 2 (poor choice) is "Ignore the alarm of others and continue moving forward."

vectors in the two spaces more aligned for "good" (or relevant) action texts compared to "bad" (or irrelevant) choices, resulting in a higher interaction function output (Q-function value).

4.1.2 Learning the DRRN: Back propagation

To learn the DRRN, we use the "experience-replay" strategy [103], which uses a fixed exploration policy to interact with the environment to obtain a sample trajectory. Then, we randomly sample a transition tuple (s_k, a_k, r_k, s_{k+1}), compute the temporal difference error for sample k:

$$d_k = r_k + \gamma \max_a Q(s_{k+1}, a; \Theta_{k-1}) - Q(s_k, a_k; \Theta_{k-1}),$$

Algorithm 1 Learning algorithm for DRRN

Initialize replay memory $\mathcal{D}$ to capacity $|\mathcal{D}|$.

Initialize DRRN with small random weights.

Initialize game simulator and load dictionary.

for $episode = 1, \ldots, M$ **do**

Restart game simulator.

Read raw state text and a list of action text from the simulator, and convert them to representation s_1 and $a_1^1, a_1^2, \ldots, a_1^{|\mathcal{A}_1|}$. $a_1^i \in \mathcal{A}_1$.

for $t = 1, \ldots, T$ **do**

Compute $Q(s_t, a_t^i; \Theta)$ for the list of actions using DRRN forward activation.

Select an action a_t based on probability distribution $\pi(a_t = a_t^i | s_t)$

Execute action a_t in simulator

Observe reward r_t. Read the next state text and the next list of action texts, and convert them to representation s_{t+1} and $a_{t+1}^1, a_{t+1}^2, \ldots, a_{t+1}^{|\mathcal{A}_{t+1}|}$.

Store transition $(s_t, a_t, r_t, s_{t+1}, \mathcal{A}_{t+1})$ in $\mathcal{D}$.

end for

Sample random mini-batch of transitions $\{(s_k, a_k, r_k, s_{k+1}, A_{k+1}) | k = 1, \cdots, B\}$ from $\mathcal{D}$.

Set $y_k = \begin{cases} r_k & \text{if } s_{k+1} \text{ is terminal} \\ r_k + \gamma \max_{a \in A_{k+1}} Q(s_{k+1}, a; \Theta)) & \text{otherwise} \end{cases}$

Perform a gradient descent step on $\sum_k (y_k - Q(s_k, a_k; \Theta))^2$ with respect to the network parameters Θ (Section 4.1.2). Back-propagation is performed only for a_k even though there are $|\mathcal{A}_k|$ actions at time k.

end for

and update the model according to the recursions:

$$W_{v,k} = W_{v,k-1} + \eta_k d_k \cdot \frac{\partial Q(s_k, a_k; \Theta_{k-1})}{\partial W_v} \tag{4.7}$$

$$b_{v,k} = b_{v,k-1} + \eta_k d_k \cdot \frac{\partial Q(s_k, a_k; \Theta_{k-1})}{\partial b_v} \tag{4.8}$$

for $v \in \{s, a\}$. Expressions for $\frac{\partial Q}{\partial W_v}$, $\frac{\partial Q}{\partial b_v}$ and other algorithm details are given in Section 4.1.3. Random sampling essentially scrambles the trajectory from experience-replay into a "bag-of-transitions", which has been shown to avoid oscillations or divergence and achieve faster convergence in Q-learning [119]. Since the models on the action side share the same parameters, models associated with all actions are effectively updated even though the back propagation is only over one action. We apply back propagation to learn how to pair the text strings from the reward signals in an end-to-end manner. The representation vectors for the state-text and the action-text are automatically learned to be aligned with each other in the text embedding space from the reward signals. A summary of the full learning algorithm is given in Algorithm 1. In the algorithm table, M is the number of episodes, T is the length of an episode, B is the size of a mini-batch for computing the gradient, t is the time index in an episode, while k denotes the index of a sample in the mini-batch. In the actual implementation each iteration we generate multiple episodes of experience and then perform multiple epochs of gradient descent in order to speed up with parallel computing.

Figure 4.2 illustrates learning with an inner product interaction function. We used Principal Component Analysis (PCA) to project the 100-dimension last hidden layer representation (before the inner product) to a 2-D plane. The vector embeddings start with small values, and after 600 episodes of experience-replay training, the embeddings are very close to the converged embedding (4000 episodes). The embedding vector of the optimal action (Action 1) converges to a positive inner product with the state embedding vector, while Action 2 converges to a negative inner product.

4.1.3 Detailed Forward and Backward Formulas for DRRN

Let $h_{l,s}$ and $h_{l,a}$ denote the l-th hidden layer for state and action side neural networks, respectively. For state side, $W_{l,s}$ and $b_{l,s}$ denote the linear transformation weight matrix and bias vector between the $(l-1)$-th and l-th hidden layers. For actions side, $W_{l,a}$ and $b_{l,a}$ denote the linear transformation weight matrix and bias vector between the $(l-1)$-th and l-th hidden layers. The DRRN has L hidden layers on each side.

For the case where the interaction function is an inner product, the forward and backward updates are as follows:

Forward:

$$h_{1,s} = f(W_{1,s}s_t + b_{1,s}) \tag{4.9}$$

$$h^i_{1,a} = f(W_{1,a}a^i_t + b_{1,a}), \quad i = 1, 2, 3, ..., |\mathcal{A}_t| \tag{4.10}$$

$$h_{l,s} = f(W_{l-1,s}h_{l-1,s} + b_{l-1,s}), \quad l = 2, 3, ..., L \tag{4.11}$$

$$h^i_{l,a} = f(W_{l-1,a}h^i_{l-1,a} + b_{l-1,a}), \quad i = 1, 2, 3, ..., |\mathcal{A}_t|, l = 2, 3, ..., L \tag{4.12}$$

$$Q(s_t, a^i_t) = h'_{L,s}h^i_{L,a} \tag{4.13}$$

where $f(\cdot)$ is the nonlinear activation function at the hidden layers, which is chosen as $\tanh(x) = (1 - \exp(-2x))/(1 + \exp(-2x))$, and $\mathcal{A}_t$ denotes the set of all actions at time t. $h'_{L,s}$ denotes the transpose of $h_{L,s}$. If using interaction functions other than the inner product, Equation 4.13 will change (so as the backward formula).

Backward:

Note we only back propagate for actions that are actually taken. More formally, let a_t be action the DRRN takes at time t, and denote

$$\Delta = [Q(s_t, a_t) - (r_t + \gamma \max_{a \in \mathcal{A}_{t+1}} Q(s_{t+1}, a))]^2/2. \tag{4.14}$$

Denote $\delta_{l,s} = \delta b_{l,s} = \partial Q/\partial b_{l,s}$, $\delta_{l,a} = \delta b_{l,a} = \partial Q/\partial b_{l,a}$, and we have (by following chain rules):

$$\delta Q = \frac{\partial \Delta}{\partial Q} = Q(s_t, a_t) - (r_t + \gamma \max_a Q(s_{t+1}, a)) \tag{4.15}$$

$$\begin{cases} \delta_{L,s} = \delta Q \cdot h_{L,a} \odot (1 - h_{L,s}) \odot (1 + h_{L,s}) \\ \delta_{l-1,s} = W'_{l,s}\delta_{l,s} \odot (1 - h_{l-1,s}) \odot (1 + h_{l-1,s}), \quad l = 2, 3, ..., L \end{cases} \tag{4.16}$$

$$\begin{cases} \delta_{L,a} = \delta Q \cdot h_{L,s} \odot (1 - h_{L,a}) \odot (1 + h_{L,a}) \\ \delta_{l-1,a} = W'_{l,a}\delta_{l,a} \odot (1 - h_{l-1,a}) \odot (1 + h_{l-1,a}), \quad l = 2, 3, ..., L \end{cases} \tag{4.17}$$

$$\begin{cases} \delta W_{1,s} = \partial Q / \partial W_{1,s} = \delta_{1,s} \cdot s'_t \\ \delta W_{l,s} = \partial Q / \partial W_{l,s} = \delta_{l,s} \cdot h'_{l-1,s}, \quad l = 2, 3, ..., L \end{cases} \tag{4.18}$$

$$\begin{cases} \delta W_{1,a} = \partial Q / \partial W_{1,a} = \delta_{1,a} \cdot a'_t \\ \delta W_{l,a} = \partial Q / \partial W_{l,a} = \delta_{l,a} \cdot h'_{l-1,a}, \quad l = 2, 3, ..., L \end{cases} \tag{4.19}$$

where $\odot$ denotes element-wise Hadamard product. For the cases where other interaction functions are used, the first lines in Equation 4.16 and 4.17 will change.

4.2 DRRN on text games

We evaluate the DRRN with the two text games described in Chapter 3: a deterministic text game task called "Saving John" and a larger-scale stochastic text game called "Machine of Death" from a public archive.

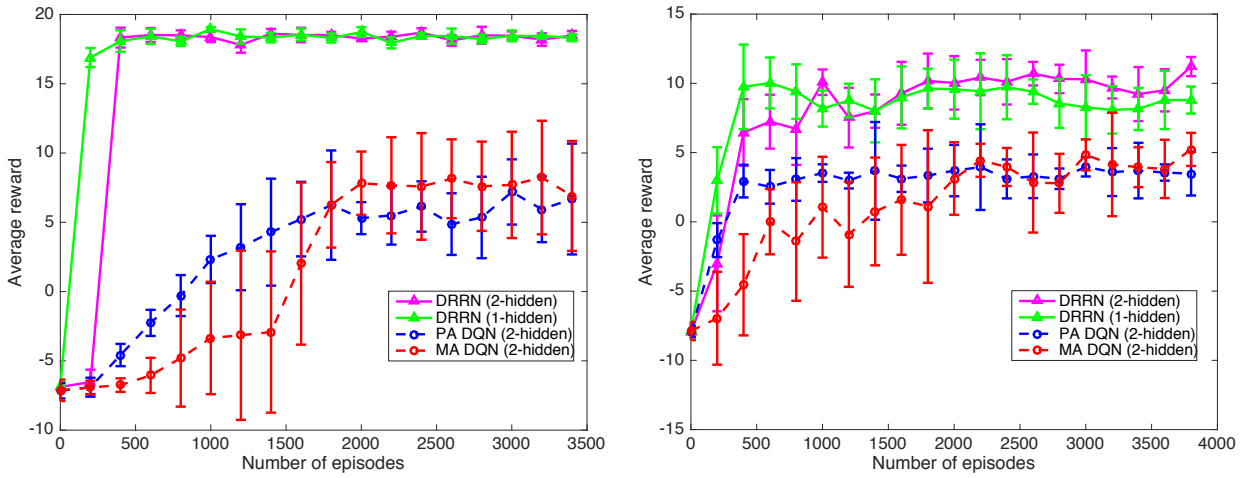
In "Saving John" all actions are choice-based, for which the mapping from text strings to a_t are clear. In "Machine of Death", when actions are hypertext, the actions are substrings of the state. In this case s_t is associated with the full state description, and a_t are given by the substrings without any surrounding context. "Saving John" is a relatively short game and stops after finite steps. For the text game "Machine of Death", we restrict an episode to be no longer than 500 steps. For text input, we use bag-of-words as features, with different vocabularies for the state side and action side.

4.2.1 Experiment setup

We apply DRRNs with both 1 and 2 hidden layer structures. In most experiments, we use dot-product as the interaction function and set the hidden dimension to be the same for each hidden layer in both state and action spaces. We compare DRRNs with 20, 50 and 100-dimension hidden layers and build learning curves during experience-replay training. The learning rate is constant: $\eta_t = 0.001$. In testing, as in training, we apply softmax selection. We record average final rewards as performance of the model.

The DRRN is compared to multiple baselines: a linear model, two max-action DQNs (MA DQN) ($L = 1$ or 2 hidden layers), and two per-action DQNs (PA DQN) (again, $L = 1, 2$). All baselines use the same Q-learning framework with different function approximators to predict $Q(s_t, a_t)$ given the current state and actions. For the linear and MA DQN baselines, the input is the same text-based state and action vector representation as in the DRRN, and the number of outputs is equal to the maximum number of actions. When there are fewer actions than the maximum, the highest scoring available action is used. For the linear model, there is no hidden layer so results are irrelevant to hidden dimension. The PA DQN baseline takes each pair of state-action texts as input, and generates a corresponding Q-value, similar to the DRRN. In terms of number of parameters in the weight matrices, the DRRN uses significantly fewer parameters than the MA DQN, and is on par with the PA DQN.

We use softmax selection, which is widely applied in practice, to trade-off exploration vs. exploitation. Specifically, for each experience-replay, we first generate 200 episodes of data (about 3K tuples in "Saving John" and 16K tuples in "Machine of Death") using the softmax selection rule, where we set $\alpha = 0.2$ for the first game and $\alpha = 1.0$ for the second game. The α is picked according to an estimation of range of the optimal Q-values. We then shuffle the generated data tuples (s_t, a_t, r_t, s_{t+1}) update the model as described in Section 4.1.2. The model is trained with multiple epochs for all configurations, and is evaluated after each experience-replay. The discount factor γ is set to 0.9. For DRRN and all baselines, network weights are initialized with small random values. To prevent algorithms from "remembering" state-action ordering and make choices based on action wording, each time the algorithm/player reads text from the simulator, we randomly shuffle the list of actions.[1] This will encourage the algorithms to make decisions based on the understanding of the texts that describe the states and actions.



(a) Game 1: "Saving John"

(b) Game 2: "Machine of Death"

Figure 4.3: Learning curves of the two text games. All systems use $L = 2$ and 100-dimensional hidden layers.

4.2.2 Performance

In Figure 4.3, we show the learning curves of different models, where the dimension of the hidden layers in the DQNs and DRRN are all set to 100. The error bars are obtained by running 5 independent experiments. The proposed methods and baselines all start at about the same performance (roughly -7 average rewards for "Saving John", and roughly -8 average rewards for "Machine of Death"), which is the random guess policy. After around 4000 episodes of experience-replay training, all methods converge. The DRRN converges much faster than the other three baselines and achieves a higher average reward. We hypothesize this is because the DRRN architecture is better at capturing relevance between state text and action text. The faster convergence for "Saving John" may be due to the smaller observation space and/or the deterministic nature of its state transitions (in contrast to the stochastic transitions in the other game).

[1]When in a specific state, the simulator presents the possible set of actions in random order, i.e. they may appear in a different order the next time a player is in this same state.

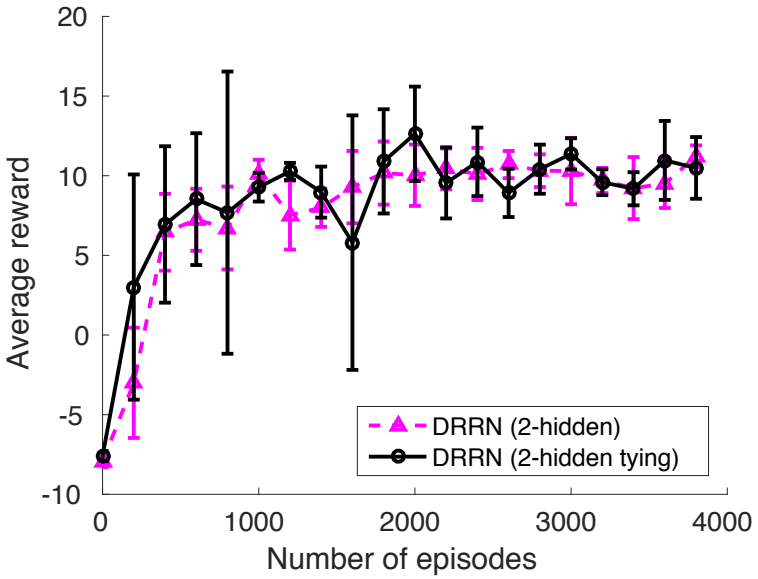


Figure 4.4: Learning curves of shared state-action embedding vs. proposed DRRN in "Machine of Death"

We also train DRRN with shared state and action embedding, on "Machine of Death". This is done by tying the parameter on the state side and the action side. The learning curve is shown in Figure 4.4. For the first 1000 episodes, parameter tying gives faster convergence, but learning curve also has high variance and unstable.

The final performance (at convergence) for both baselines and proposed methods are shown in Tables 4.1 and 4.2. We set the same hidden dimension for both state and action embeddings and test for different model sizes with 20, 50, and 100 dimensions in the hidden layers. The DRRN performs consistently better than all baselines, and often with a lower variance. For "Machine of Death", due to the complexity of the underlying state transition function, we cannot compute the exact optimal policy score. To provide more insight into the performance, we averaged scores of 8 human players for initial trials (novice) and after gaining experience, yielding scores of -5.5 and 16.0, respectively. The experienced players do outperform our algorithm. The converged performance is higher with two hidden layers for all models. However, deep models also converge more slowly than their 1 hidden layer

Eval metric	Average reward		
hidden dimension	20	50	100
Linear	4.4 (0.4)		
PA DQN ($L = 1$)	2.0 (1.5)	4.0 (1.4)	4.4 (2.0)
PA DQN ($L = 2$)	1.5 (3.0)	4.5 (2.5)	7.9 (3.0)
MA DQN ($L = 1$)	2.9 (3.1)	4.0 (4.2)	5.9 (2.5)
MA DQN ($L = 2$)	4.9 (3.2)	9.0 (3.2)	7.1 (3.1)
DRRN ($L = 1$)	17.1 (0.6)	18.3 (0.2)	18.2 (0.2)
DRRN ($L = 2$)	18.4 (0.1)	18.5 (0.3)	**18.7** (0.4)

Table 4.1: The final average rewards and standard deviations on "Saving John".

Eval metric	Average reward		
hidden dimension	20	50	100
Linear	3.3 (1.0)		
PA DQN ($L = 1$)	0.9 (2.4)	2.3 (0.9)	3.1 (1.3)
PA DQN ($L = 2$)	1.3 (1.2)	2.3 (1.6)	3.4 (1.7)
MA DQN ($L = 1$)	2.0 (1.2)	3.7 (1.6)	4.8 (2.9)
MA DQN ($L = 2$)	2.8 (0.9)	4.3 (0.9)	5.2 (1.2)
DRRN ($L = 1$)	7.2 (1.5)	8.4 (1.3)	8.7 (0.9)
DRRN ($L = 2$)	9.2 (2.1)	10.7 (2.7)	**11.2** (0.6)

Table 4.2: The final average rewards and standard deviations on "Machine of Death".

versions, as shown for the DRRN in Figure 4.3.

Besides an inner-product, we also experimented with more complex interaction functions: a) a bilinear operation with different action side dimensions; and b) a non-linear neural network using the concatenated state and action space embeddings as input and trained in an end-to-end fashion to predict Q values. For different configurations, we fix the state side embedding to be 100 dimensions and vary the action side embedding dimensions. The

Eval metric	Average reward		
hidden dimension	20	50	100
Inner product	9.2 (2.1)	10.7 (2.7)	**11.2** (0.6)
Bilinear	8.9 (1.8)	9.7 (1.9)	**11.3** (1.9)
Concatenation + NN	8.7 (1.4)	9.3 (0.4)	9.8 (2.5)

Table 4.3: The final average rewards and standard deviations on "Machine of Death", using DRRN ($L = 2$) with different interaction functions. "Bilinear" refers to computing Q-values using a bilinear operation, with a 100-dimension state vector and different embedding dimensions for the action side. "Concatenation + NN" refers to computing the Q function using a NN with the concatenation of state and action embeddings as input.

bilinear operation gave similar results, but the concatenation input to a DNN degraded performance. The results are shown in Table 4.3. Similar behaviors have been observed on a different task [108].

To provide some insight into the model, we show examples of state-action pairs in the two text games, in Table 4.4 and Table 4.5. Those values make sense in their game context, respectively.

4.2.3 Actions with paraphrased descriptions

To investigate how our models handle actions with "unseen" natural language descriptions, we had two people paraphrase all actions in the game "Machine of Death" (used only in the testing phase), except a few single-word actions whose synonyms are out-of-vocabulary (OOV). The word-level OOV rate of paraphrased actions is 18.6%, and standard 4-gram BLEU score between the paraphrased and original actions is 0.325. The resulting 153 paraphrased action descriptions are associated with 532 unique state-action pairs. The full set of paraphrases is given in `https://github.com/jvking/text-games/blob/master/simulators/machineofdeath_originalActions.txt` and `https://github.com/jvking/text-games/blob/master/simulators/machineofdeath_paraphrasedActions.txt`.

State	Actions (with Q values)
A wet strand of hair hinders my vision and I'm back in the water. Sharp pain pierces my lungs. How much longer do I have? 30 seconds? Less? I need to focus. A hand comes into view once more.	I still don't know what to do. (-8.981) Reach for it. (18.005)
"Me:" Hello Sent: today "Cherie:" Hey. Can I call you? Sent: today	Reply "I'll call you" (14.569) No (-9.498)
"You don't hold any power over me. Not anymore." Lucretia raises one eyebrow. The bar is quiet. "I really wish I did my hair today." She twirls a strand. "I'm sorry," "Save it." //Yellow Submarine plays softly in the background.// "I really hate her." "Cherie? It's not her fault." "You'll be sorry," "Please stop screaming."	I laugh and she throws a glass of water in my face. (16.214) I look away and she sips her glass quietly. (-7.986)
My dad left before I could remember. My mom worked all the time but she had to take care of her father, my grandpa. The routine was that she had an hour between her morning shift and afternoon shift, where she'd make food for me to bring to pops. He lived three blocks away, in a house with red steps leading up to the metal front door. Inside, the stained yellow wallpaper and rotten oranges reeked of mold. I'd walk by myself to my grandfather's and back. It was lonely sometimes, being a kid and all, but it was nothing I couldn't deal with. It's not like he abused me, I mean it hurt but why wouldn't I fight back? I met Adam on one of these walks. He made me feel stronger, like I can face anything.	Repress this memory (-8.102) Why didn't I fight back? (10.601) Face Cherie (14.583)

Table 4.4: Q values (in parentheses) for state-action pair from "Saving John", using trained DRRN. High Q-value actions are more cooperative actions thus more likely leading to better endings

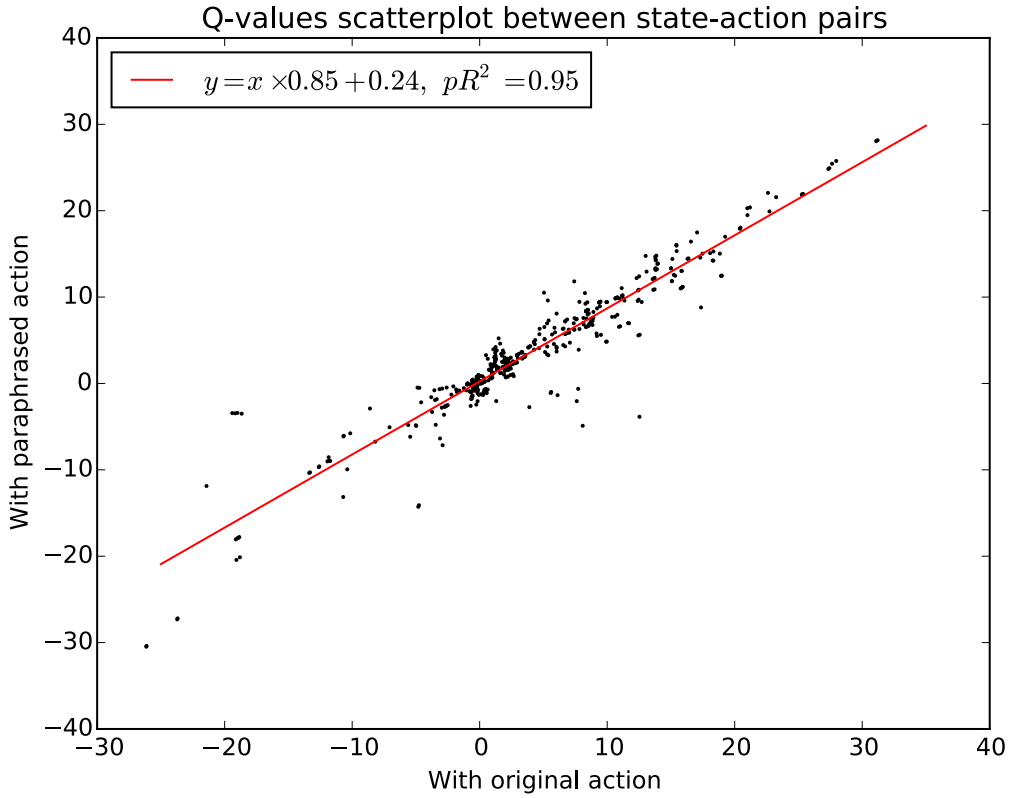


Figure 4.5: Scatterplot and strong correlation between Q-values of paraphrased actions versus original actions

We apply a well-trained 2-layer DRRN model (with hidden dimension 100), and predict Q-values for each state-action pair with fixed model parameters. Figure 4.5 shows the correlation between Q-values associated with paraphrased actions versus original actions. The predictive R-squared is 0.95, showing a strong positive correlation. We also run Q-value correlation for the NN interaction and $pR^2 = 0.90$. For baseline MA-DQN and PA-DQN, their corresponding pR^2 is 0.84 and 0.97, respectively, indicating they also have some generalization ability. This is confirmed in the paraphrasing-based experiments too, where the test reward on the paraphrased setup is close to the original setup. This supports the claim that deep learning is useful in general for this language understanding task, and our findings show that a decoupled architecture most effectively leverages that approach.

In Table 4.6 we provide examples with predicted Q-values of original descriptions and paraphrased descriptions. We also include alternative action descriptions with in-vocabulary words that will lead to positive / negative / irrelevant game development at that particular state. Table 4.6 shows actions that are more likely to result in good endings are predicted with high Q-values. This indicates that the DRRN has some generalization ability and gains a useful level of language understanding in the game scenario.

We use the baseline models and the DRRN model trained with the original action descriptions for "Machine of Death", and test on paraphrased action descriptions with the underlying state transition mechanism unchanged. The only change to the game interface is that during testing, every time the player reads the actions from the game simulator, it reads the paraphrased descriptions and performs selection based on these paraphrases. Since the texts in test time are "unseen" to the player, a good model needs to have some level of language understanding, while a naive model that memorizes all unique action texts in the original game will do poorly. The results for these models are shown in Table 4.7. All methods have a slightly lower average reward in this setting (10.5 vs. 11.2 for the original actions), but the DRRN still gives a high reward and significantly outperforms other methods. This shows that the DRRN can generalize well to "unseen" natural language descriptions of actions.

4.3 DRRN for Reddit thread tracking $K = 1$

In this section, we tested the DRRN on the Reddit thread tracking task with $K = 1$. This is another application with natural language action space, and formulation of this problem is discussed in Section 3.2. For text preprocessing, we remove punctuation and lowercase capital letters. Since $K = 1$, each action a_t^i is selecting one comment c_t^i. The state s_t is a one-thread sequence of history comments that were selected. For each state s_t and action a_t^i, we use a bag-of-words representation with the most frequent 5K vocabulary in all networks. We compare the DRRN with two DQN baselines, MA-DQN and PA-DQN, on 5 subreddits (askscience / askmen / todayilearned / askwomen / politics). The actions in MA-DQN are

random shuffled to avoid temporal information. We set the window size $N = 10$.

For the Q-learning agent, fully-connected neural networks are used for text embedding, and each network has $L = 2$ hidden layers, each with hidden dimension 20. In our preliminary experiments on askscience, hidden dimension 20 works well while 50 or 100 leads to overtraining. ϵ-greedy is used in exploration-exploitation, and we keep $\epsilon = 0.1$ throughout training and testing. The discount factor $\gamma = 0.9$ and experience replay memory is set to 10,000. For each experience replay, 500 episodes are generated and stored in a first-in-first-out fashion. Minibatch gradient descent is used with a batch size of 100, and we use the Adam optimization scheduler [82] with learning rate $\eta_t = 0.00001$. In the DRRN, we use an inner product as the pairwise interaction function.

In Table 4.8, we show test performance for DRRN and baselines together with results for the random policy and the oracle upper bound. The results represent the average (and standard deviation) over 5 runs, where each run is an average over 1000 episodes. Upperbounds are estimated by searching through each discussion tree to find K max Karma discussion threads (overlapped comments are counted only once).[2] We see a significant and consistent gain in using the DRRN over all baseline performances. In Reddit popularity prediction and tracking, PA-DQN performs better than MA-DQN. However, all these models are still far from the oracle upper bound, which is expected. This experiment serves as additional evidence that DRRN works well in a natural language action space.

According to our experiments, myopically picking direct replies with the highest (oracle) karma only gives 50% to 60% of the upperbound if we do a long term search in an offline constructed tree. Testing on the askscience subreddit, the performance of an automatic myopic policy (supervised learning with karma scores as labels) is significantly worse than reinforcement learning using the same feature set, which supports our claim that long-term dependency indeed matters. An example is given in Figure 3.3. This serves as a justification that reinforcement learning is an appropriate approach for modeling popularity of a

[2]This upper bound may not be attainable in a real-time setting, because the greedy search is according to tree structures rather than time.

discussion thread.

State	Actions (with Q values)
Peak hour ended an hour or so ago, alleviating the feeling of being a tinned sardine that?s commonly associated with shopping malls, though there are still quite a few people busily bumbling about. To your left is a fast food restaurant. To the right is a UFO catcher, and a poster is hanging on the wall beside it. Behind you is the one of the mall's exits. In front of you stands the Machine. You're carrying 4 dollars in change.	fast food restaurant (1.094) the Machine (3.708) mall's exits (0.900) UFO catcher (2.646) poster (1.062)
You lift the warm mug to your lips and take a small sip of hot tea.	Ask what he was looking for. (3.709) Ask about the blood stains. (7.488) Drink tea. (5.526) Wait. (6.557)
As you move forward, the people surrounding you suddenly look up with terror in their faces, and flee the street.	Ignore the alarm of others and continue moving forward. (-21.464) Look up. (16.593)
Are you happy? Is this what you want to do? If you didn't avoid that sign, would you be satisfied with how your life had turned out? Sure, you're good at your job and it pays well, but is that all you want from work? If not, maybe it's time for a change.	Screw it. I'm going to find a new life right now. It's not going to be easy, but it's what I want. (23.205) Maybe one day. But I'm satisfied right now, and I have bills to pay. Keep on going. (One minute) (14.491)
You slam your entire weight against the man, making him stumble backwards and drop the chair to the ground as a group of patrons race to restrain him. You feel someone grab your arm, and look over to see that it?s Rachel. Let's get out of here, she says while motioning towards the exit. You charge out of the bar and leap back into your car, adrenaline still pumping through your veins. As you slam the door, the glove box pops open and reveals your gun.	Grab it and hide it in your jacket before Rachel can see it. (21.885) Leave it. (1.915)

Table 4.5: Q values (in parentheses) for state-action pair from "Machine of Death", using trained DRRN

	Text (with predicted Q-values)
State	As you move forward, the people surrounding you suddenly look up with terror in their faces, and flee the street.
Actions in the original game	Ignore the alarm of others and continue moving forward. (-21.5) Look up. (16.6)
Paraphrased actions (not original)	Disregard the caution of others and keep pushing ahead. (-11.9) Turn up and look. (17.5)
Positive actions (not original)	Stay there. (2.8) Stay calmly. (2.0)
Negative actions (not original)	Screw it. I'm going carefully. (-17.4) Yell at everyone. (-13.5)
Irrelevant actions (not original)	Insert a coin. (-1.4) Throw a coin to the ground. (-3.6)

Table 4.6: Predicted Q-value examples

Eval metric	Average reward		
hidden dimension	20	50	100
PA DQN ($L = 2$)	0.2 (1.2)	2.6 (1.0)	3.6 (0.3)
MA DQN ($L = 2$)	2.5 (1.3)	4.0 (0.9)	5.1 (1.1)
DRRN ($L = 2$)	7.3 (0.7)	8.3 (0.7)	**10.5 (0.9)**

Table 4.7: The final average rewards and standard deviations on the paraphrased revision of the game "Machine of Death".

models	askscience	askmen	todayilearned	askwomen	politics
random policy	95.3 (1.0)	36.8 (1.0)	112.8 (2.9)	37.6 (0.9)	43.3 (0.9)
MA-DQN (L=2)	171.4 (5.2)	40.7 (0.8)	142.2 (6.2)	39.5 (0.5)	48.7 (1.7)
PA-DQN (L=2)	359.1(19.8)	42.4 (2.3)	204.4 (6.7)	44.6 (1.9)	54.2 (2.7)
DRRN (L=2)	**473.9** (9.3)	**43.5** (0.8)	**210.8** (9.1)	**46.4** (1.5)	**60.1** (2.9)
upper bound	1665.8 (33.0)	334.6 (8.2)	1425.5 (29.5)	271.1 (11.3)	557.0 (15.7)

Table 4.8: A performance comparison (across different subreddits) with $N = 10, K = 1$. All systems have hidden dimension 20.

Chapter 5

DRL WITH A COMBINATORIAL ACTION SPACE

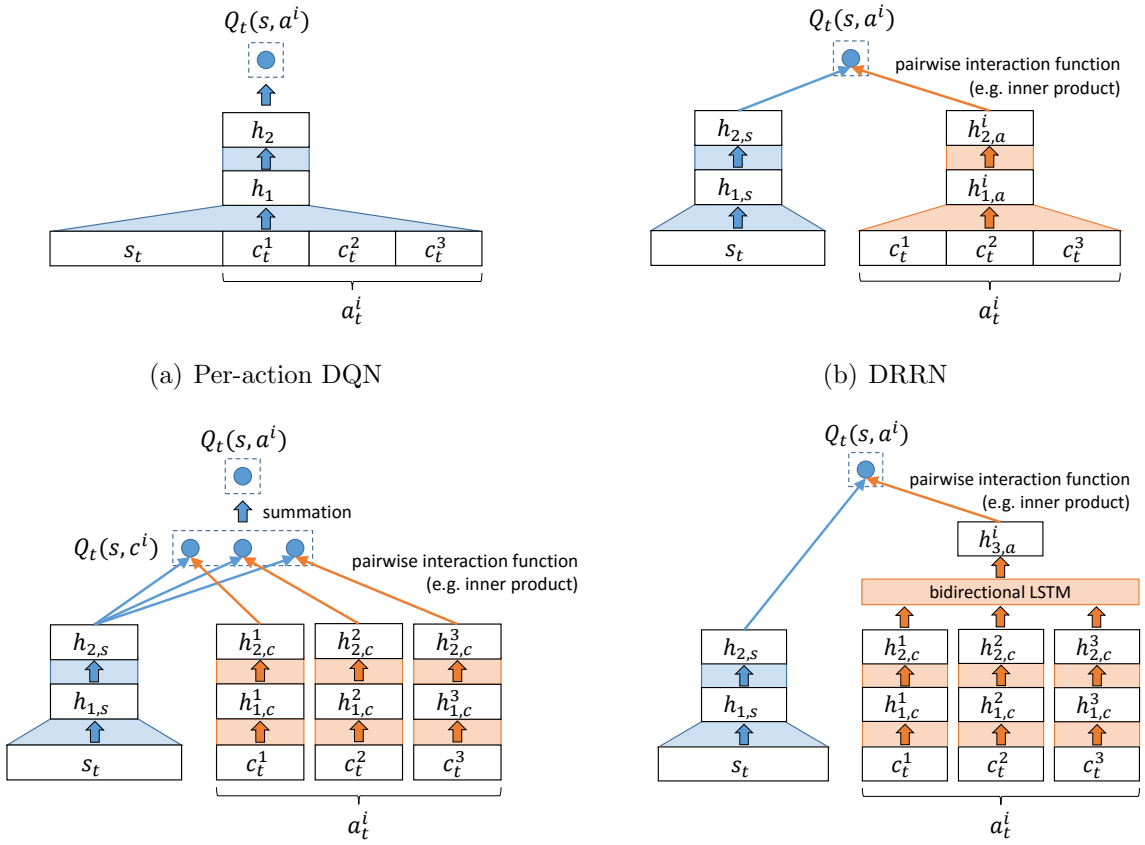
The problem of a combinatorial natural language action space arises in many applications. For example, in real-time news feed recommendation, a user may want to read diverse topics of interest, and an action (i.e. recommendation) from the computer agent would consist of a set of news articles that are not all similar in topics [193]. In advertisement placement, an action is a selection of several ads to display, and bundling with complementary products might receive a higher click-through-rate than displaying all similar popular products.

In this chapter, we investigate a combinatorial natural language action space with the Reddit task, i.e. predicting and tracking popular discussion threads on Reddit. In contrast to the case discussed in the previous chapter, the agent can choose K comments to track out of the recent N observed (vs. only choosing one), so there are $\binom{N}{K}$ possible actions. There are two challenges associated with a combinatorial action space. One is the development of a Q-function framework for estimating the long-term reward when the K comments (sub-actions) are inter-dependent. The other is the potentially high computational complexity, due to evaluating Q over every possible pair of (s_t, a_t^i).

The work in this chapter was published in two venues. Proposing novel architectures for better modeling the Q-function is presented in [59], and using two-stage Q-learning for reducing search complexity is presented in [62].

5.1 Model Q-function

With the real-time setting, it is clear that action a_t will affect the next state s_{t+1} and furthermore the future expected reward. The action a_t consists of K comments (sub-actions), making modeling Q-values $Q(s_t, a_t)$ difficult. In the previous chapter, it is shown that the



(a) Per-action DQN

(b) DRRN

(c) DRRN-Sum

(d) DRRN-BiLSTM

Figure 5.1: Different deep Q-learning architectures

Deep Reinforcement Relevance Network (DRRN, Figure 5.1(b)), i.e. two separate deep neural networks for modeling state embedding and action embedding, performs better than per-action DQN (PA-DQN in Figure 5.1(a)), as well as other DQN variants for dealing with natural language action spaces.

Our baseline models include Linear, PA-DQN and DRRN. We concatenate the K sub-actions/comments to form the action representation. The Linear and PA-DQN (Figure 5.1(a)) take as input a concatenation of state and action representations, and model a single Q-value $Q(s_t, a_t)$ using linear or DNN function approximations. The DRRN consists of a pair of DNNs, one for the state-text embedding and the other for action-text embeddings,

which are then used to compute $Q(s_t, a_t)$ via a pairwise interaction function (Figure 5.1(b)).

One simple alternative approach by utilizing this combinatorial structure is to compute an embedding for each sub-action c_t^i. We can then model the value in picking a particular sub-action, $Q(s_t, c_t^i)$, through a pairwise interaction between the state and this sub-action. $Q(s_t, c_t^i)$ represents the expected accumulated future rewards by including this sub-action. The agent then greedily picks the top-K sub-actions with highest values to achieve the highest $Q(s_t, a_t)$. In this approach, we are assuming the long-term rewards associated with sub-actions are independent of each other. More specifically, greedily picking the top-K sub-actions is equivalent to maximizing the following action-value function:

$$Q(s_t, a_t) = \sum_{i=1}^{K} Q(s_t, c_t^i) \tag{5.1}$$

while satisfying (3.1). We call this proposed method DRRN-Sum, and its architecture is shown in Figure 5.1(c). Similarly as in DRRN, we use two networks to embed state and actions separately. However, for different sub-actions, we keep the network parameters tied. We also use the same top layer dimension and the same pairwise interaction function for all sub-actions.

In the case of a linear additive interaction, such as an inner product or bilinear operation, Equation (5.1) is equivalent to computing the interaction between the state embedding and an action embedding, where the action embedding is obtained linearly by summing over K sub-action embeddings. When sub-actions have strong correlation, this independence assumption is invalid and can result in a poor estimation of $Q(s_t, a_t)$. For example, most people are interested in the total information stored in the combined action a_t. Due to content redundancy in the sub-actions $c_t^1, c_t^2, \cdots, c_t^K$, we expect $Q(s_t, a_t)$ to be smaller than $\sum_i Q(s_t, c_t^i)$.

To come up with a general model for handling a combinatorial action-value function, we further propose the DRRN-BiLSTM (Figure 5.1(d)). In this architecture, we use a DNN to generate an embedding for each comment. Then a Bidirectional Long Short-Term Memory [49] is used to combine a sequence of K comment embeddings. As the Bidirectional LSTM

has a larger capacity, we expect it will capture more details on how the embeddings for the sub-actions combine into an action embedding. Note that both of our proposed methods (DRRN-Sum and DRRN-BiLSTM) can handle a varying value of K, while for the DQN and DRRN baselines, we need to use a fixed K in training and testing.

5.2 Reduce search complexity

In the case of deep Q-learning, most of the time has been spent on the forward-pass from $\binom{N}{K}$ actions to $\binom{N}{K}$ Q-values. For back-propagation, since we only need to back-propagate one particular action the agent has chosen, complexity is not affected by the combinatorial action space.

One solution to reduce computational complexity is simply to randomly pick a fixed number, say m candidate actions, and perform a max operation. While this is widely used in the reinforcement learning literature, it is problematic in our application because the large and highly skewed action space makes it likely that good actions are missed. Here we propose to use two-stage Q-learning for reducing search complexity using coarse-to-fine models. More specifically, we can rewrite the max operation as:

$$\max_{a_t \in \mathcal{A}_t} Q_2(s_t, a_t) \approx \max_{a_t \in \mathcal{B}_t} Q_2(s_t, a_t)$$

where

$$\mathcal{B}_t = \arg\max_{a_t \in \mathcal{A}_t}^{m} Q_1(s_t, a_t) \tag{5.2}$$

where $\arg\max^m_{a_t \in \mathcal{A}_t}$ means picking the top-m actions from the whole action set $\mathcal{A}_t$.

In the case of Q_1 being DRRN-Sum, we can rewrite $Q_1(s_t, a_t)$ as:

$$Q_1(s_t, a_t) = \sum_{i=1}^{K} Q_0(s_t, c_t^i) = \sum_{i=1}^{K} q_t^i$$

which is simplified by precomputing sub-action value $q_t^i = Q_0(s_t, c_t^i)$, $i = 1, \cdots, N$. Q_0 is the simple DRRN introduced in Chapter 4.

To further elaborate, the idea is to use a function Q_1 to first perform a quick but rough ranking of a_t^i. The second function Q_2, which can be more sophisticated, is used to rerank the top-m candidate actions. This ensures that all comments are explored, and at the same time, the architecture can be sophisticated enough to capture detailed dependencies between sub-actions, such as information redundancy. In our experiments, we pick Q_1 to be DRRN-Sum and Q_2 to be DRRN-BiLSTM. While the independence assumption on sub-action interdependency is too strong, the DRRN-Sum model is relatively easy to train. Since the parameters on the action side are tied for different sub-actions, we can train a DRRN with $K = 1$ and then apply the model for each pair of (s_t, c_t^i). This will result in N sub-action Q-values $Q_0(s_t, c_t^i), i = 1, 2, \cdots, N$. Thus computing Equation 5.2 is equivalent to retrieving the top m from $\binom{N}{K}$ values. Thus, we avoid the huge computational cost of first generating $\binom{N}{K}$ actions from N sub-actions, then applying a general Q-function approximation to evaluate all of them. We used the DRRN with $K = 1$ from Chapter 4 and then copy the parameters to DRRN-Sum, which can be used to evaluate the full action space. The whole two-stage Q framework is summarized in Algorithm 2.

5.3 Experiments

5.3.1 Datasets and Experimental Configurations

Our data consists of 5 subreddits (askscience, askmen, todayilearned, worldnews, nfl) with diverse topics and genres. We report the random policy performances and oracle upper bound performances on the test set (averaged over 10,000 episodes) in Table 5.1 and Table 5.2. Upper bounds are estimated by greedily searching through each discussion tree to find K max karma discussion threads (overlapped comments are counted only once). The upper bound performances are obtained using stabilized karma scores and offline constructed tree structure. This upper bound may not be attainable in real-time setting. The mean and standard deviation are obtained by 5 independent runs, as in the $K = 1$ experiments. In all our experiments we set $N = 10$ as in Chapter 4.

Subreddit	**Random**	**Upper bound**
askscience	321.3 (7.0)	2109.0 (16.5)
askmen	132.4 (0.7)	651.4 (2.8)
todayilearned	390.3 (5.7)	2679.6 (30.1)
worldnews	205.8 (4.5)	1853.4 (44.4)
nfl	237.1 (1.4)	1338.2 (13.2)
askwomen	128.9 (2.3)	574.0 (10.9)
politics	149.3 (1.7)	967.7 (13.6)

Table 5.1: Mean and standard deviation of random and upper-bound performance (with $N = 10, K = 3$) across different subreddits.

K	**Random**	**Upper bound**
2	201.0 (2.1)	1991.3 (2.9)
3	321.3 (7.0)	2109.0 (16.5)
4	447.1 (10.8)	2206.6 (8.2)
5	561.3 (18.8)	2298.0 (29.1)

Table 5.2: Mean and standard deviation of random and upper-bound performance on askscience, with $N = 10$ and $K = 2, 3, 4, 5$.

In terms of the Q-learning agent, fully-connected neural networks are used for text embeddings. The network has $L = 2$ hidden layers, each with 20 nodes, and model parameters are initialized with small random numbers. ϵ-greedy is used for exploration-exploitation, and we keep $\epsilon = 0.1$ throughout online training and testing. We pick the discount factor $\gamma = 0.9$. During online training, we use experience replay [102] and the memory size is set to 10,000 tuples of $(s_t, a_t, r_{t+1}, s_{t+1})$. For each experience replay, 500 episodes are generated and stored in a first-in-first-out fashion, and multiple epochs are trained for each model. Minibatch stochastic gradient descent is implemented with a batch size of 100. The learning rate is kept constant: $\eta_t = 0.000001$.

Two series of experiments are conducted. First, we assess the Q function alternatives using random sampling in the search with $m = 10$. Then, the two-stage Q function is evaluated. The proposed BiLSTM Q function is compared with four baseline models: Linear, per-action DQN (PA-DQN), DRRN, and DRRN-Sum. For both Linear and PA-DQN, the state and comments are concatenated as an input. For the DRRN, DRRN-Sum, and DRRN-BiLSTM, we use an inner product as the pairwise interaction function between the state and action vectors.

5.3.2 Results Using Random Sampling

In Figure 5.2 we provide learning curves of different models on the askscience subreddit during online learning. In this experiment, we set $N = 10, K = 3$. Each curve is obtained by averaging over 3 independent runs, and the error bars are also shown. All models start with random performance, and converge after approximately 15 experience replays. The DRRN-Sum converges as fast as baseline models, with better converged performance. DRRN-BiLSTM converges slower than other methods, but with the best converged performance.

After we train all the models on the training set, we fix the model parameters and apply (deploy) on the test set, where the models predict which action to take but no reward is shown until evaluation. The test performance is averaged over 1000 episodes, and we report mean and standard deviation over 5 independent runs.

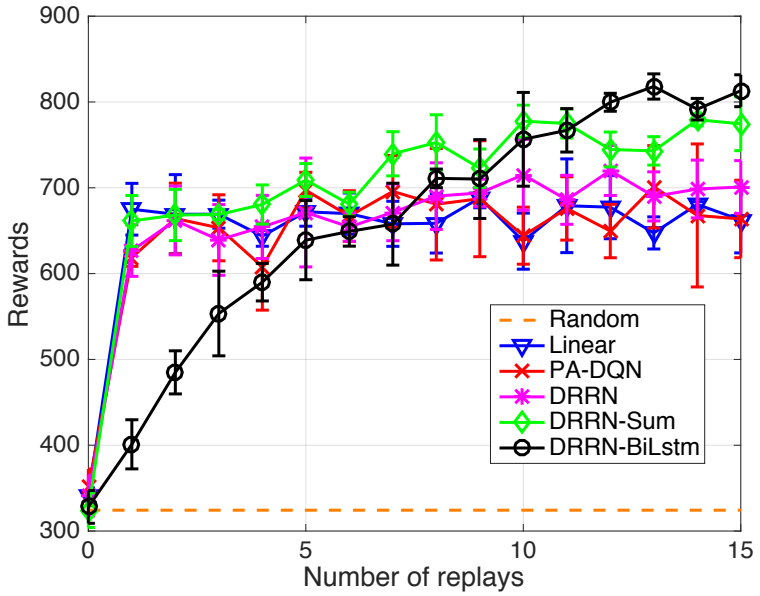


Figure 5.2: Learning curves of baselines and proposed methods on "askscience"

On askscience, we try multiple settings with $N = 10$, $K = 2, 3, 4, 5$ and the results are shown in Table 5.3. Both DRRN-Sum and DRRN-BiLSTM consistently outperform baseline methods. The DRRN-BiLSTM performs better with larger K, probably due to the greater chance of redundancy in combining more sub-actions.

We also perform online training and test across different subreddits. With $N = 10, K = 3$, the test performance gains over the linear baseline are shown in Figure 5.3. Again, the test performance is averaged over 1000 episodes, and we report mean and standard deviation over 5 independent runs. The findings are consistent with those for askscience. Since different subreddits may have very different karma scores distributions and language style, this suggests the algorithms apply to different text genres.

In actual model deployment, a possible scenario is that users may have different requests. For example, a user may ask the agent to provide $K = 2$ discussion threads on one day, due to limited reading time, and ask the agent to provide $K = 5$ discussion threads on the other day. For the baseline models (Linear, PA-DQN, DRRN), we will need to train separate

K	Linear	PA-DQN	DRRN	DRRN-Sum	DRRN-BiLSTM
2	553.3 (2.8)	556.8 (14.5)	553.0 (17.5)	569.6 (18.4)	**573.2 (12.9)**
3	656.2 (22.5)	668.3 (19.9)	694.9 (15.5)	704.3 (20.1)	**711.1 (8.7)**
4	812.5 (23.4)	818.0 (29.9)	828.2 (27.5)	829.9 (13.2)	**854.7 (16.0)**
5	861.6 (28.3)	884.3 (11.4)	921.8 (10.7)	942.3 (19.1)	**980.9 (21.1)**

Table 5.3: On askscience, average karma scores and standard deviation of baselines and proposed methods (with $N = 10$)

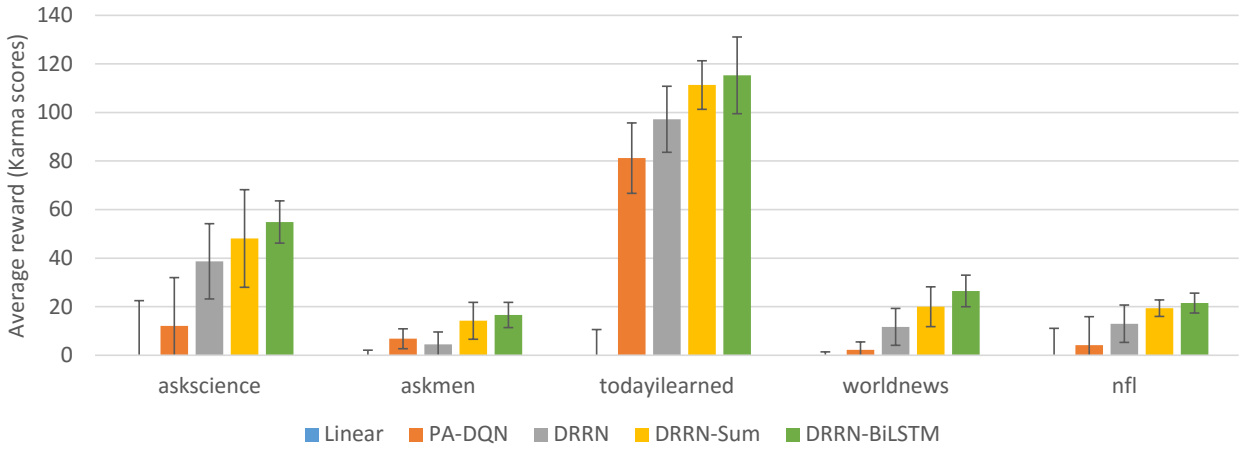


Figure 5.3: Average karma score gains over the linear baseline and standard deviation across different subreddits (with $N = 10, K = 3$).

models for different K's. The proposed methods (DRRN-Sum and DRRN-BiLSTM), on the other hand, can easily handle a varying K. To test whether the performance indeed generalizes well, we train proposed models on askscience with $N = 10, K = 3$ and test them with $N = 10, K \in 2, 4, 5$, as shown in Table 5.4. Compared to the proposed models that are specifically trained for these K's (Table 5.3), the generalized test performance indeed degrades, as expected. However, in many cases, our proposed methods still outperform all three baselines (Linear, PA-DQN and DRRN) that are trained specifically for these K's. This shows that the proposed methods can generalize to varying K's even if it is trained on

K	DRRN-Sum	DRRN-BiLSTM
2	538.5 (18.9)	**551.2 (10.5)**
4	819.1 (14.7)	**829.9 (11.1)**
5	921.6 (15.6)	**951.3 (15.7)**

Table 5.4: On askscience, average karma scores and standard deviation of proposed methods trained with $K = 3$ and test with different K's

State text (partially shown)
Are there any cosmological phenomena that we strongly suspect will occur, but the universe just isn't old enough for them to have happened yet?
Comments (sub-actions) (partially shown)
[1] White dwarf stars will eventually stop emitting light and become black dwarfs. [2] Yes, there are quite a few, such as: White dwarfs will cool down to black dwarfs.

Table 5.5: An example state and its sub-actions

a particular value of K.

In Table 5.5, we show an anecdotal example with state and sub-actions. The two sub-actions are strongly correlated and have redundant information. By combining the second sub-action compared to choosing just the first sub-action alone, DRRN-Sum and DRRN-BiLSTM predict 86% and 26% relative increase in action-value, respectively. Since these two sub-actions are highly redundant, we hypothesize DRRN-BiLSTM is better than DRRN-Sum at capturing interdependency between sub-actions.

5.3.3 Two-stage Q-learning

In this subsection we study the effect of two-stage Q-learning. On askscience, we try multiple settings with $K = 2, 3, 4, 5$ and the results are shown in Table 5.6. We compare the proposed two-stage Q-learning with two single-stage Q-learning baselines. The first baseline, following

$\mathcal{B}_t$	random	all	DRRN-Sum
Q_2	DRRN-BiLSTM	DRRN-Sum	DRRN-BiLSTM
K=2	573.2 (12.9)	663.3 (8.7)	**676.9 (5.5)**
K=3	711.1 (8.7)	793.1 (8.1)	**833.9 (5.7)**
K=4	854.7 (16.0)	964.5 (12.0)	**987.1 (12.1)**
K=5	980.9 (21.1)	1099.4 (15.9)	**1101.3 (13.8)**

Table 5.6: A performance comparison (across different K's on askscience subreddit)

the method in Section 5.3.2, uses a random subsampling approach to obtain $\mathcal{B}_t$ (with $m = 10$) and takes the max over them using DRRN-BiLSTM. The second baseline uses DRRN-Sum and explores the whole action space.[1] The proposed two-stage Q-learning uses DRRN-Sum for picking a $\mathcal{B}_t$ and DRRN-BiLSTM for reranking. We observe a large improvement by switching from "random" to "all", showing that exploring the entire action space is important in this task. There is a consistent gain by using two-stage Q-learning instead of a single-stage Q with DRRN-Sum. This shows that using a more sophisticated value function for reranking also helps with performance.

In Table 5.7, we compare two-stage Q-learning with the two baselines across different subreddits, with $N = 10, K = 3$. The findings are consistent with those for askscience. Since different subreddits may have very different karma score distributions and language style, our results suggest that the algorithm applies well to different community interaction styles.

[1]We also tried to compare training a single DRRN-BiLSTM with the whole action space. However this is intractable both in runtime and experience replay memory usage. Training DRRN-BiLSTM with a larger subspace (sampled with $m = 20$ and $m = 50$) and testing on the whole action space, however, degrades performance compared to the two-stage Q-learning approach. We hypothesize this is due to mismatch in training and testing.

$\mathcal{B}_t$	random	all	DRRN-Sum
Q_2	DRRN-BiLSTM	DRRN-Sum	DRRN-BiLSTM
askscience	711.1 (8.7)	793.1 (8.1)	**833.9 (5.7)**
askmen	139.0 (3.6)	142.5 (2.3)	**148.0 (5.5)**
todayilearned	606.9 (15.8)	679.4 (11.4)	**697.9 (9.4)**
askwomen	135.0 (1.3)	145.9 (2.4)	**149.6 (3.3)**
politics	177.9 (3.3)	180.6 (6.3)	**204.7 (4.2)**

Table 5.7: A performance comparison (across different subreddits) with $K = 3$, $N = 10$

5.3.4 Cost Comparisons

During testing, we compare runtime of the DRRN-BiLSTM Q-function with different $\mathcal{B}_t$, simulating over 10,000 episodes with $N = 10$ and $K = 2, 3, 4, 5$. We use 4-core 2.30GHz CPU with a Tesla K80 GPU and results are shown in Figure 5.4. For fair comparisons, we chose Q_2=DRRN-BiLSTM and tested with different approaches for $\mathcal{B}_t$. The two-stage Q-function approach takes slightly longer time compared to random subsampling, but is significantly faster than exploring the whole action space. More specifically, the search time for the random selection and the two-stage Q-function are similar, both nearly constant for different K. Using two-stage Q the test runtime is reduced by 6× for $K = 3$ and 11× for $K = 5$ comparing to exploring the whole action space. Note that training DRRN-BiLSTM with the whole action space is intractable, so we just used a subspace-trained DRRN-BiLSTM model for testing. This however achieves worse performance compared to the two-stage Q probably due to mismatch in training and testing.

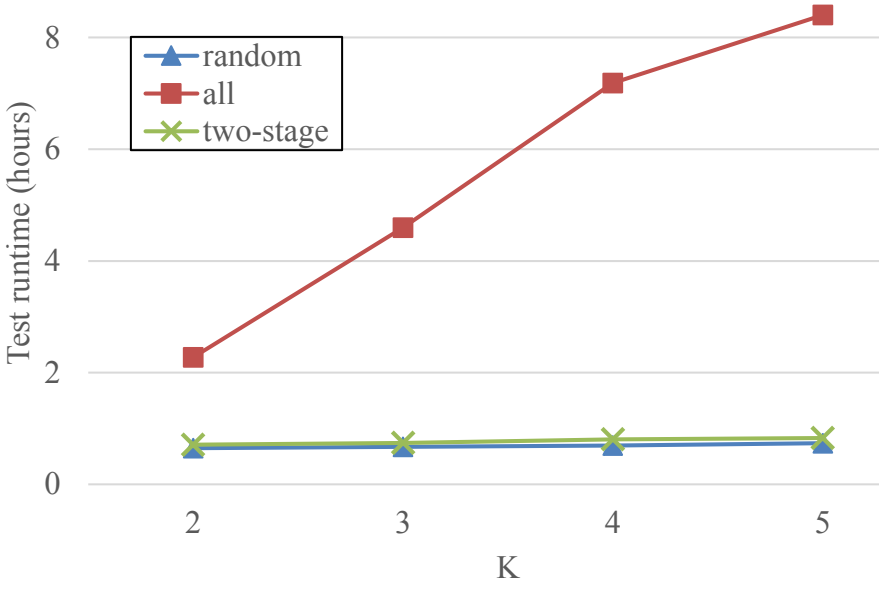


Figure 5.4: Test runtime of $N = 10$, Q_2=DRRN-BiLSTM and different approaches for $\mathcal{B}_t$

Algorithm 2 Two-stage Q-learning (Q_1: DRRN-Sum, Q_2: DRRN-BiLSTM)

Initialize Reddit popularity prediction environment and load dictionary.

Initialize DRRN $Q_0(s_t, c_t^i; \Theta_1)$ with small random weights and train. The DRRN-Sum $Q_1(s_t, a_t; \Theta_1) = Q_1(s_t, \{c_t^1, c_t^2, \cdots, c_t^K\}; \Theta_1) = \sum_{i=1}^{K} Q_0(s_t, c_t^i; \Theta_1)$

Initialize replay memory $\mathcal{D}$ to capacity $|\mathcal{D}|$.

for $episode = 1, \ldots, M$ **do**

Randomly pick a discussion tree.

Read raw state text and a list of sub-action text from the simulator, and convert them to representation s_1 and $c_{1,1}, c_{1,2}, \ldots, c_{1,N}$.

Compute $q_{1,j} = Q_0(s_1, c_{1,j}; \Theta_1)$ for the list of sub-actions using DRRN forward pass.

For each $a_1 \in \mathcal{A}_1$, form value of $Q_1(s_1, a_1; \Theta_1) = \sum_{i=1}^{K} Q_0(s_1, c_1^i; \Theta_1) = \sum_{i=1}^{K} q_1^i$.

Keep a list of top m actions $\mathcal{B}_1 = [a_1^1, a_1^2, \cdots, a_1^m]$; each a_1^i consists of K sub-actions.

for $t = 1, \ldots, T$ **do**

Compute $Q_2(s_t, a_t^i; \Theta_2), i = 1, 2, \cdots, m$ for $\mathcal{B}_t$, the list of top m actions using DRRN-BiLSTM forward pass.

Select an action a_t based on policy $\pi(a_t = a_t^i | s_t)$ derived from Q_2. Execute a_t.

Observe reward r_{t+1}. Read the next state text and the next list of sub-action texts, and convert them to representation s_{t+1} and $c_{t+1,1}, c_{t+1,2}, \ldots, c_{t+1,N}$.

Compute $q_{t+1,j} = Q_0(s_{t+1}, c_{t+1,j}; \Theta_1)$ for the list of sub-actions using DRRN.

For each $a_{t+1} \in \mathcal{A}_{t+1}$, form value of $Q_{t+1}(s_{t+1}, a_{t+1}; \Theta_1) = \sum_{i=1}^{K} Q_0(s_{t+1}, c_{t+1}^i; \Theta_1) = \sum_{i=1}^{K} q_{t+1}^i$.

Keep a list of top m actions $\mathcal{B}_{t+1} = [a_{t+1}^1, a_{t+1}^2, \cdots, a_{t+1}^m]$, where each a_{t+1}^i consists of K sub-actions.

Store transition $(s_t, a_t, r_{t+1}, s_{t+1}, \mathcal{B}_{t+1})$ in $\mathcal{D}$.

end for

if during training **then** Run Algorithm 3.

end if

end for

Algorithm 3 Mini-batch gradient descent

Sample random mini batch of transitions $\{(s_k, a_k, r_{k+1}, s_{k+1}, \mathcal{B}_{k+1}) | k = 1, \cdots, B\}$ from $\mathcal{D}$.

Set $y_k = \begin{cases} r_{k+1} & \text{if } s_{k+1} \text{ is terminal} \\ r_{k+1} + \gamma \max_{a \in \mathcal{B}_{k+1}} Q_2(s_{k+1}, a; \Theta_2)) & \text{otherwise} \end{cases}$

Perform a gradient descent step on $\sum_k (y_k - Q_2(s_k, a_k; \Theta_2))^2$ with respect to the network parameters Θ_2.

Chapter 6

INCORPORATING EXTERNAL KNOWLEDGE IN DRL

This chapter describes our contribution in incorporating external knowledge for predicting popular Reddit threads. Section 6.1 introduces background literatures on using external knowledge as a source of information. Section 6.2 describes the framework for incorporating external knowledge, in the form of a collection of unstructured world news documents, as well as temporal context extracted in the process of tracking and predicting popular discussion threads. Section 6.3 presents experimental results. Incorporating external world knowledge consistently improves over different configurations, and learns to attend to different parts of knowledge depending on the discussion community. Adding temporal features on top of best configurations gives mixed performances for different domains.

6.1 Background

People have used "external knowledge" to mean a variety of things in NLP, including linguistic resources such as WordNet [190], structured knowledge bases [43, 16, 101], Wikipedia [79, 186] and other unstructured text resources [123, 189, 2, 17, 20]. In general, the motivation for using external knowledge is to improve coverage and/or robustness for tasks such as question answering [43, 16, 79, 189], information extraction [186, 123], and knowledge base population [101, 2]. In this thesis, we are primarily interested in external knowledge represented by unstructured text, so some of this work is potentially relevant to our approach. However, our objective is for this knowledge to provide global social context for identifying popular discussion comments, so the text sources are different and metadata is potentially useful.

In question answering, knowledge representation and reasoning plays a central role [43,

16]. In the TREC Question Answering track 2005, MIT CSAIL's entries focused on using external general-knowledge sources such as Wikipedia in factoid question answering [79]. In [101], two fundamentally different approaches of using the web data for question answering are presented and compared. In the federated approach, techniques for managing semistructured data are used and the web resources are treated as databases. In the distributed approach, web data is viewed as a collection of unstructured text with redundancy. In human-computer dialog systems such as a chatbot, information retrieval approach that can leverage unstructured documents is used for responding to utterances [189]. In information extraction, large plain-text collections such as newspaper documents are used for generating patterns and extracting tuples (objects and their relations) in an iterative way [2]. Relation-specific training examples are generated by matching Infobox (a set of tuples summarizing the key attributes of the subject) on Wikipedia, to train an open information extractor [186]. Our work differs from these studies in that the knowledge is used here as context rather than as an information source, specifically to augment the state in reinforcement learning. In terms of retrieving text, tf-idf (term-frequency inverse-document-frequency) [144] is especially useful in measuring semantic similarity. Recently, automatically learned embeddings for computing relevance are also studied in question answering [159] and machine translation [108].

Our approach to leveraging external knowledge is similar to a memory network [159], which uses an attention mechanism and provides an automatic method for learning an end-to-end relevance function. In our work, we also leverage metadata (time, popularity) for computing attention mechanism.

In reinforcement learning, Narasimhan et al. [123] present a framework of acquiring and incorporating external evidence to improve extraction accuracy in domains where the amount of training data is scarce. There are also efforts to utilize natural language external knowledge in domain-specific tasks. In computer games, an approach to language grounding is presented which automatically interprets text in the manual and uses domain knowledge extracted from the text to improve game performance [17]. In an interactive computer

operating system environment, Branavan et al. address the task of mapping high-level instructions to sequences of commands using policy gradient [20]. These studies are the most closely related to our work, but they do not involve a natural language action space. In addition, they focus on facts and instructions in their retrieving external knowledge, while in our task the agent learns incorporating external knowledge for social context.

6.2 Framework

6.2.1 External world knowledge

The mechanism to incorporate external language knowledge into decision making assumes that the agent will keep track of a memory space that helps with decision making, and when a new state comes, the agent refers to this external knowledge and picks relevant resources to help with decision making. The external knowledge used here is comprised of other discussions about world events, assuming that these events and public reactions to them impact how people react to discussion comments.

The specific architecture we propose is illustrated in Figure 6.1. The agent keeps a growing collection of documents. That is, at each time t, the external knowledge contains documents from worldnews that appear before time t. Every time the agent reads the state information from the environment (Step 6 and Step 13 in Algorithm 2), it performs a lookup operation in external knowledge in its memory. This external knowledge is an evolving collection of documents from the worldnews subreddit. A document, in our case, is the post plus top-5 popular comments in a worldnews discussion tree. We use an attention mechanism that produces a probability distribution over the entire external knowledge resource. This weight vector is computed by considering a set of features measuring the relevance between the current state and each document in the collection. More specifically, we consider the following three types of relevance:

- Timing features: When users express their opinions on a website such as Reddit, it is likely they are referring to more recent news events. We use two indicator features to

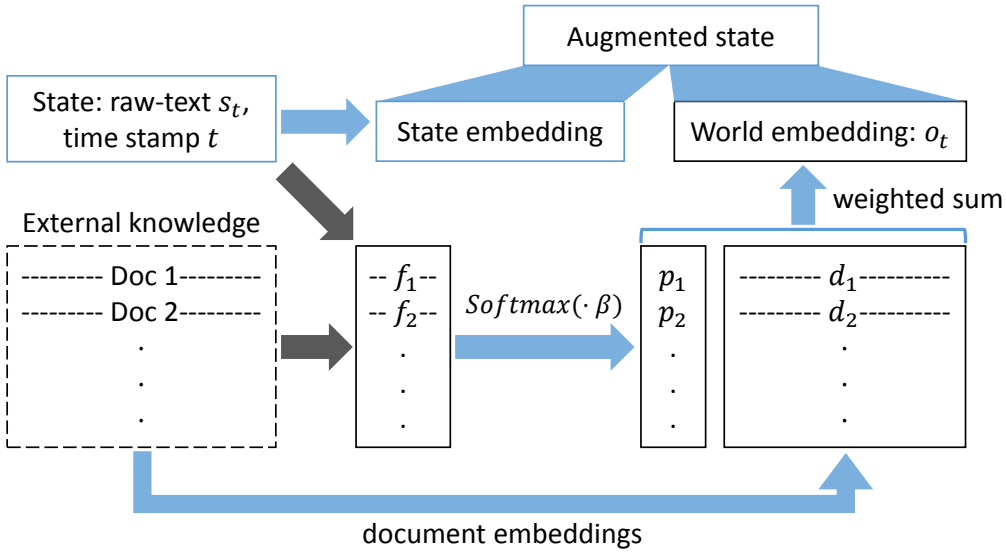


Figure 6.1: Incorporating external knowledge to augment a state-side representation with an attention mechanism. The attention features $\{f_1, f_2, \cdots\}$ depend on the state and time stamp, helping the agent learn to pay different attention to external knowledge given different states. The shaded blue parts are learned end-to-end within reinforcement learning.

represent whether a document from the external knowledge is within the past 24 hours, or the past 7 days relative to the time of the new state. We denote these features as $\mathbb{1}_{\text{day}}$ and $\mathbb{1}_{\text{week}}$, respectively.

- Semantic similarity: We use the standard tf-idf representation [143] of the state (comments tracked so far) and the documents and compute cosine similarity scores between the current state and each document in the external knowledge, as a measure for semantic relevance. We denote this semantic similarity as u_{semantic}, and $u_{\text{semantic}} \in [-1, 1]$.

- Popularity: For Reddit posts/comments, we use karma score as a measure for popularity, assuming that more people are familiar with high popularity topics. To compensate for the range difference compared to other relevance measures, we normalize karma scores. Specifically, we sum the karma scores of the post and top-5 comments, and then normalize the sum by dividing by the highest score in the external knowledge, so

the feature values fall in the range $[0, 1]$. We denote this normalized popularity score as $u_{\text{popularity}}$. Unlike in Fang et al. [40], the summed karma scores do not follow a Zipfian distribution, so we do not use quantization or any nonlinear transformation.

For each state the agent extracts the above features for each document in the external knowledge, and forms a four-dimensional feature vector $f = [\mathbb{1}_{\text{day}}, \mathbb{1}_{\text{week}}, u_{\text{semantic}}, u_{\text{popularity}}]$. The attention weights are then computed as a linear combination followed by a softmax over the entire external knowledge:

$$\boldsymbol{p} = \text{Softmax}([\mathbf{1}_{\text{day}}, \mathbf{1}_{\text{week}}, \boldsymbol{u}_{\text{semantic}}, \boldsymbol{u}_{\text{popularity}}] \cdot \boldsymbol{\beta})$$

where the Softmax operates over the collection of documents and $\boldsymbol{p}$ has dimension equal to the number of documents. Note in our experimental setting, the softmax applies for only documents that exist before the new comments appear, and this simulates a "real-time" dynamic external knowledge resource. The attention weights $\boldsymbol{p}$ are then multiplied with document embeddings $\{d_i\}$ to form a vector representation (embedding) of "world" knowledge:

$$o = \sum_i p_i d_i$$

The world embedding is concatenated with the original state embedding to enrich understanding of the environment.

Empirically, this architecture also demonstrates advantages, as presented in the experimental section. First, the proposed architecture that augments the state side representation with the world embedding leads to better performance and can also be easily extended to other deep Q-learning variants. Second, the effects of the four relevance features are domain specific, i.e. some features help more in one domain but not so much in another. Our model learns $\boldsymbol{\beta}$ in an end-to-end manner, and accounts for advantages of different features in different domains.

6.2.2 Temporal context

In this subsection, we incorporate temporal context in predicting popular Reddit discussion threads, and we consider timing as another form of knowledge other than language. More specifically, we consider two types of temporal context, absolute timing and relative timing. Absolute timing includes hour of the day (24-dimension one-hot encoding) and day of the week (7-dimension one-hot encoding). Relative timing includes time since when the post was posted, time since the parent comment, time since the last comment to its parent, rate of commenting (window length between the 10 comment window). All relative timing features are in hours.

Since timing features and text features are very distinct two types of features. We treat them on top of previously introduced models. At each time step in the tracking process, the agent sees a set of comments as sub-actions. For each comment, the agent extracts their temporal context and concatenates with a predicted Q-value from the previous model (e.g. DRRN with external knowledge in $K = 1$ case, or two-stage Q with external knowledge in combinatorial case). Then a linear function approximation is used for computing a final Q-value by considering both temporal context and text features. In the case of a combinatorial action space, we choose summary statistics (i.e. sample averages) to be the temporal features representing a set of sub-action temporal context. Specifically, when an action consists of K sub-actions, each sub-action will provide a timing feature, and we average those K vectors as the timing feature for this action.

6.3 Experiments

6.3.1 External knowledge configuration

We first study the effect of incorporating external knowledge, without considering the combinatorial action space. More specifically, we set $K = 1$ and use the simple DRRN. Each action is to pick a comment $\{c_t^1\}$ from $\mathcal{C}_t$ to track. We utilize 9.88k posts from the worldnews subreddit as our external knowledge source.

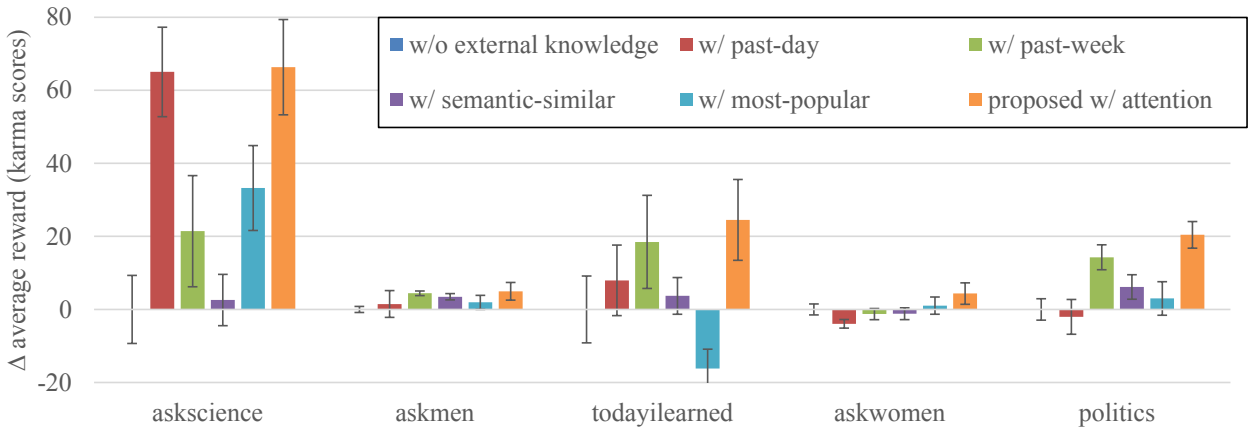


Figure 6.2: Absolute gain in performance over a DRRN without external knowledge associated with different ways of incorporating external knowledge, for 5 different subreddits

For comparison, we experiment with a baseline DRRN without any external knowledge, as described in Section 4.3. We also construct a baseline DRRN with hand-crafted rules for picking documents from external knowledge. Those rules include: i) documents within the past-day, ii) documents within the past-week, iii) the 10 semantically most similar documents, and iv) the 10 most popular documents.

6.3.2 Experimental results using worldnews

Results showing the absolute performance gains associated with different ways of incorporating external knowledge for different subreddits over a baseline DRRN (without any external knowledge) are given in Figure 6.2. The experimental results show that the DRRN using a learned attention mechanism to retrieve relevant knowledge outperforms all the rule-based configurations of DRRNs with external knowledge, and significantly outperforms the DRRN baseline that does not use external knowledge. For askmen and askwomen the absolute gains are small, but in fact the relative gains are similar to todayilearned as shown next. Also we observe that different relevance features have different impact across subreddits. For example, for askscience, past-day documents have higher impact than past-week documents,

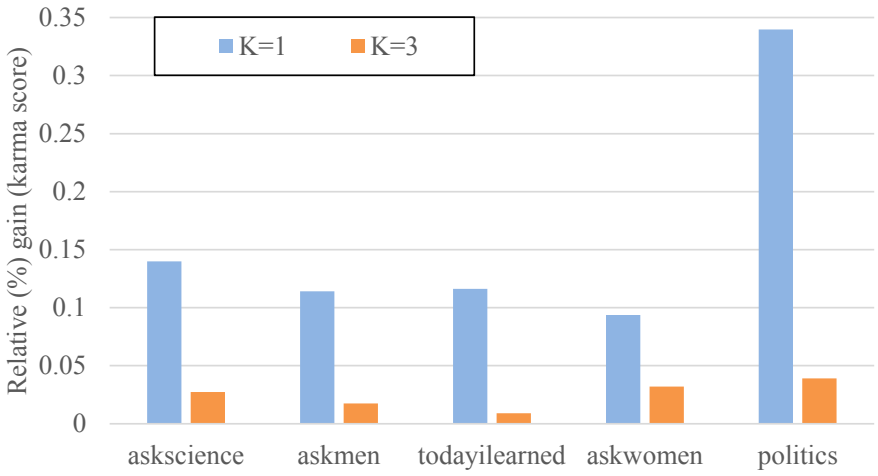


Figure 6.3: Relative (%) gain in performance from incorporating external knowledge across 5 different subreddits, for $N = 10$.

while for politics past-week documents are more important. The most-popular documents actually have a negative effect for todayilearned, mainly because those are documents which are most popular throughout the entire history, while todayilearned discussions value information about recent events. Therefore, the attention mechanism seems to learn to rely on proper features to retrieve useful knowledge for the needs of different domains. In principle, since we are concatenating the world embedding to obtain an augmented state representation, the result should not get worse. We hypothesize this is due to overfitting and use of mismatched documents, as in the most-popular setting for todayilearned.

In Figure 6.3, we present the results for the model using attention, but this time showing relative gain. The figure shows effects of incorporating external knowledge with $N = 10$ for both $K = 1$ and the combinatorial $K = 3$ case using two-stage Q-learning. In all cases, incorporating external knowledge consistently gives a performance gain, but the benefits are much smaller for $K = 3$. This suggests that external knowledge is mainly useful for finding the very top thread, but less important if you are following multiple threads. The relative gain for politics (34%) is substantially larger than for the other subreddits, because worldnews tends to include a lot of political news. The lower impact on askmen and askwomen is

state	Would it be possible to artificially create an atmosphere like Earth has on **Mars**?	Does our **sun** have any unique features compared to any other **star**?
top-1	Ultimate Reality TV: A Crazy Plan for a **Mars** Colony - It might become the mother of all reality shows. Fully 704 candidates are soon to begin competing for a trip to **Mars** to establish a colony there.	**Star** Wars: Episode VII begins filming in UAE desert. This can't possibly be a modern **Star** Wars movie! I don't see a green screen in sight! Ya, it's more like Galaxy news.
top-2	'Alien thigh bone' on **Mars**: Excitement from alien hunters at 'evidence' of extraterrestrial life. Mars likely never had enough oxygen in its atmosphere and elsewhere to support more complex organisms.	African Pop **Star** turns white (and causes controversy) with new line of skin whitening cream. I would like to see an unshopped photo of her in natural lighting.
top-3	The Gaia (General Authority on Islamic Affairs) and the UAE (United Arab Emirates) have issued a fatwa on people living on **mars**, due to the religious reasoning that there is no reason to be there.	Dwarf planet discovery hints at a hidden Super Earth in solar system - The body, which orbits the **sun** at a greater distance than any other known object, may be shepherded by an unseen planet.
least	North Korea's internet is offline; massive DDOS attack presumed.	Hong Kong democracy movement hit by 2018. The vote has no standing in law, by attempting to sabotage it, the Chinese(?) are giving it legitimacy

Table 6.1: States and documents (partial text) showing how the agent learns to attend to different parts of external knowledge

presumably because they cover more personal topics.

In Table 6.1, we show examples of most/least attended documents in the external knowledge given the state description. The documents are shortened for brevity. In the first example, the state is about a question about the atmosphere on Mars. The most-attended documents are correctly related to Mars living conditions, in various respects. The second example has the state talking about sun's features compared to other stars. Interestingly, although the agent is able to attend to top documents due to some topic word matching (e.g. sun, star), the picked documents reflect popularity more than topic relevance. In terms of popularity, both the Reality TV show in the first example and the Star Wars movie in the second example have the highest karma scores among the top-3 documents, respectively, showing the model's consideration for other relevance other than semantic similarity. The least-attended documents are totally irrelevant in both examples, as expected. Furthermore, we observe in other examples that the most-attended documents indeed change when the state changes even within a discussion thread, showing the agent's ability to adapt to different parts of the knowledge source when the topic changes.

6.3.3 Incorporating temporal context experimental results

We further experiment with adding temporal context on top of the best configurations so far on the Reddit task. For the $K = 1$ scenario, the extracted temporal features have 36 dimensions (4 relative timing features + 7 dimensions day of the week + 24 dimensions hour of the day), and are concatenated with predicted q values from a DRRN with external worldnews knowledge. Again we experiment with 5 subreddits and results are shown in Table 6.2.

From Table 6.2, timing features are helpful for most subreddits except askscience. We hypothesize that this is because askscience popularity is related more to the quality of answers in response to a scientific question. By looking at the parameters of the linear model, parameters associated with absolute timing features take similar values, indicating they are not very informative. This is expected, as our task focuses on picking a high popularity

External knowledge?	Timing?	askscience	askmen	todayilearned	askwomen	politics
N	N	473.9 (9.3)	43.5 (0.8)	210.8 (9.1)	46.4 (1.5)	60.1 (2.9)
Y	N	540.3 (13.0)	48.5 (2.4)	235.3 (11.1)	49.5 (1.8)	80.5 (3.6)
Y	Y	517.1 (7.2)	52.5 (3.8)	237.6 (3.8)	51.2 (1.2)	83.4 (5.0)

Table 6.2: Effects of timing features (across different subreddits with $K = 1$)

comment in a relatively short time window. Thus actions within that time window will have the same or neighboring absolute timing features.

We further experiment with a combinatorial action space with $K = 3$. In this case, only relative timing features are added on top of the two-stage Q with incorporating external worldnews knowledge. Results are shown in Table 6.3. Unfortunately timing features again are hurting performance on askscience. For other subreddits, we see consistent but small gains.

The small gain is surprising given that other work has shown that timing is important [76]. We hypothesize two possible reasons. One is that timing features are mainly important because high karma comments often come early, but in our task setting the window of 10 descendants effectively forces a choice of comments that arrive early. The second reason is that timing mainly filters out less important comments, which represent the vast majority of comments and therefore dominate evaluation metrics that consider all comments. In our task definition, only high karma comments substantially contribute to the score.

External knowledge?	Timing?	askscience	askmen	todayilearned	askwomen	politics
N	N	833.9 (5.7)	148.0 (5.5)	697.9 (9.4)	149.6 (3.3)	204.7 (4.2)
Y	N	856.8 (8.1)	150.6 (6.1)	704.3 (9.8)	154.4 (2.9)	212.7 (3.9)
Y	Y	839.5 (16.5)	155.6 (8.5)	706.8 (9.4)	156.6 (2.2)	216.4 (8.4)

Table 6.3: Effects of timing features (across different subreddits with $K = 3$)

Chapter 7

CONCLUSION

7.1 Summary of contributions

There has been increasing interest in applying deep reinforcement learning to a variety of problems, but only a few studies address problems with natural language state and action spaces. Our work serves as one of pioneers in this direction. In our setting, both the state and the action spaces are inherently discrete, but we learn a continuous representation of each using a new architecture for deep reinforcement learning with variants to handle problems with combinatorial natural language actions (i.e. text selection). We also formulate two new testbeds for RL: text-based games and predicting popular Reddit threads. In both tasks, the states and actions are all described by natural language so they are useful for language studies. Publicly available source codes are released and can be accessed on github.

More specifically, we first we develop a deep reinforcement relevance network (DRRN), a novel DNN architecture for handling actions described by natural language in decision-making tasks such as text games. In experiments with two text games and a discussion thread tracking task, we show that the DRRN converges faster and to a better solution for Q-learning than alternative architectures that do not use separate embeddings for the state and action spaces.

Second, we develop novel deep Q-learning architectures to better model the state-action value function with a combinatorial action space. The sub-actions are interdependent so action value function is not a simple summation of sub-action values. The proposed DRRN-BiLSTM method not only performs better than DRRN or Per-action DQN baselines across different experimental configurations and domains, but it also generalizes well for scenarios where the user can request changes in the number tracked. To explore the entire action

space while avoiding enumeration of all possible action combinations, we introduce a two-stage Q-learning framework. Experimental results show performance improvement in the task of predicting popular Reddit threads.

Third, we develop a novel architecture for incorporating unstructured external knowledge into reinforcement learning. Information from the original state is used to query the knowledge source (an evolving collection of documents corresponding to other online discussions about world events), and the state representation is augmented by the outcome of the query. This learnable attention mechanism (depending on multiple factors such as time, popularity, and topic matching) gives improved performance in the task of predicting popular Reddit threads, and gives dynamic attention conditioned on the current state. We also provide a mechanism for including temporal features in characterizing the actions but find minimal gain due to the restrictions on the action space of a short time window.

7.2 Future work

In this thesis, we follow the value function approach for reinforcement learning in natural language scenarios. This is useful when a task involves a natural language state and action space, and the agent interacts with the environment by evaluating a set of given actions and picking one of them.

An important application that involves natural language state and action spaces is human-computer dialogs. Human-computer dialog systems aim at providing an automated language assistant, through typed-in text or spoken language, that converses with human beings and performs certain tasks. The task could be either a specific domain-dependent task such as booking flight tickets, providing information given a knowledge base, or providing social interactions and entertainment such as chatbots. In a human-computer dialog system, the state is usually the conversation history, or some extracted form with belief updates from context. The action is a response generated to be replied to the user, or a dialog act from a carefully designed set of agent intents in a specific domain. Our thesis methods could be applied when the state is in natural language form and the action space is a set of candidate

responses. However, they are not directly applicable if we want to generate a natural language response. To do that, we could follow a policy-based approach. Instead of focusing on modeling an action value function, we can directly model the policy, i.e. conditional probability of taking an action given the state. In natural language processing, this is essentially language modeling conditioned on context. The policy could be further refined within the reinforcement learning framework. In the future the distinction between task-oriented dialog systems and social/chat bots might be more obscure, and a unified framework for handling general purpose dialog will be preferred. At present there are two research approaches. A group of researchers start with modular systems and try to scale up to more domains, while another stream of research starts with an end-to-end approach and try to improve upon existing algorithms.

One might also wish to explore other forms of knowledge, such as a structured knowledge base. In Chapter 6, we present an approach for incorporating a collection of unstructured documents as a source of external knowledge. We also consider timing features as other types of knowledge. In many applications a structured database or knowledge base is available, which may be beneficial to query for external information that helps with the current reinforcement learning task. In addition, the structure in the database can be utilized to speed up the query process, unlike the softmax attention approach we used with a collection of documents.

It will also be of interest to apply deep reinforcement learning to other applications that involve language, such as recommendation systems and strategic financial/business planning. In those applications, text/language features may serve as a set of complementary features to enhance performance. For example, a recommendation system usually uses multi-arm bandits modeling or collaborative filtering that considers history and co-occurrence information in action selection. However it has proven helpful to combine content (such as text) information for cold start situations [147]. In financial pricing, value iteration and function approximation are widely used. It might be helpful to also make use of NLP features in function approximation.

Deep reinforcement learning is an exciting area that attracts more and more researchers every year. This partly reflects how technologies evolve and develop. After the GMM-HMM framework was first introduced in speech processing, for many years there had been incremental algorithmic improvement to automatic speech recognition, until the realm of deep learning. By the time deep learning was proven useful in speech recognition and computer vision, researchers devoted to seeking applications where this new technique might solve other long-standing problems. This is similar to what is happening now in deep reinforcement learning. Motivated by the huge success of AlphaGo, many researchers working on reinforcement learning are interested to see if there are other applications where this algorithm can create disruptive innovation, as impactful as AlphaGo. On the other hand, every technique has limits. We will see whether deep reinforcement learning (and its variants) will eventually lead to so-called "strong artificial intelligence" or "artificial general intelligence."

BIBLIOGRAPHY

[1] E. Adams. *Fundamentals of game design*. Pearson Education, 2014.

[2] E. Agichtein and L. 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] J. Allan. *Topic detection and tracking: event-based information organization*, volume 12. Springer Science & Business Media, 2012.

[4] Tim Althoff, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. How to ask for a favor: A case study on the success of altruistic requests. *arXiv preprint arXiv:1405.3282*, 2014.

[5] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[6] E. Ammicht, A. L Gorin, and T. Alonso. Knowledge collection for natural language spoken dialog systems. In *EUROSPEECH*, 1999.

[7] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

[8] Lalit R Bahl, Peter F Brown, Peter V de Souza, and Robert L Mercer. Speech recognition with continuous-parameter hidden markov models. *Computer Speech & Language*, 2(3-4):219–234, 1987.

[9] Leemon C Baird III. *Reinforcement learning through gradient descent*. PhD thesis, US Air Force Academy, US, 1999.

[10] Roja Bandari, Sitaram Asur, and Bernardo Huberman. The pulse of news in social media: forecasting popularity. In *Proc. Int. AAAI Conf. Web and Social Media (ICWSM)*, 2012.

[11] Regina Barzilay, Kathleen R McKeown, and Michael Elhadad. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 550–557. Association for Computational Linguistics, 1999.

[12] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.(JAIR)*, 47:253–279, 2013.

[13] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[14] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[15] Dan Bohus, Alexander Rudnicky, et al. Constructing accurate beliefs in spoken dialog systems. In *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*, pages 272–277. IEEE, 2005.

[16] J. Boyd-Graber, B. Satinoff, H. He, and H. Daumé III. Besting the quiz master: Crowdsourcing incremental classification games. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1290–1301. Association for Computational Linguistics, 2012.

[17] S. Branavan, D. Silver, and R. Barzilay. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704, 2012.

[18] S.R.K. Branavan, H. Chen, L. Zettlemoyer, and R. Barzilay. Reinforcement learning for mapping instructions to actions. In *Proc. of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th IJCNLP*, pages 82–90, August 2009.

[19] 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.

[20] 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.

[21] Jianshu Chen, Ji He, Xiaodong He, Lin Xiao, Jianfeng Gao, and Li Deng. Interpreting the prediction process of a deep network constructed from supervised topic models. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 2429–2433. IEEE, 2016.

[22] Jianshu Chen, Ji He, Yelong Shen, Lin Xiao, Xiaodong He, Jianfeng Gao, Xinying Song, and Li Deng. End-to-end learning of lda by mirror-descent back propagation over a deep architecture. In *Advances in Neural Information Processing Systems*, pages 1765–1773, 2015.

[23] Justin Cheng, Lada Adamic, P. Alex Dow, Jon Kleinberg, and Jure Leskovec. Can cascades be predicted? In *Proc. Int. Conf. World Wide Web (WWW)*, 2014.

[24] Freddy Chong Tat Chua and Sitaram Asur. Automatic summarization of events from social media. In *ICWSM*. Citeseer, 2013.

[25] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[26] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of the 25th International Conference on Machine learning*, pages 160–167. ACM, 2008.

[27] Heriberto Cuayáhuitl, Simon Keizer, and Oliver Lemon. Strategic dialogue management via deep reinforcement learning. *arXiv preprint arXiv:1511.08099*, 2015.

[28] G. E Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42, 2012.

[29] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[30] Hoa Trang Dang and Karolina Owczarzak. Overview of the tac 2008 update summarization task. In *Proceedings of text analysis conference*, pages 1–16, 2008.

[31] H. Daumé III, J. Langford, and D. Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

[32] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357–366, 1980.

[33] Gerald DeJong. An overview of the frump system. *Strategies for natural language processing*, 113, 1982.

[34] L. Deng and D. Yu. Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387, 2014.

[35] B. Dhingra, L. Li, X. Li, J. Gao, Y-N Chen, F. Ahmed, and L. Deng. End-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*, 2016.

[36] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

[37] Yajuan Duan, Long Jiang, Tao Qin, Ming Zhou, and Heung-Yeung Shum. An empirical study on learning to rank of tweets. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 295–303. Association for Computational Linguistics, 2010.

[38] G. Dulac-Arnold, R. Evans, H. Van Hasselt, P. Sunehag, T. Lillicrap, and J. Hunt. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2016.

[39] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and M. Mausam. Open information extraction: The second generation. In *IJCAI*, volume 11, pages 3–10, 2011.

[40] H. Fang, H. Cheng, and M. Ostendorf. Learning latent local conversation modes for predicting community endorsement in online discussions. In *Proc. Int. Workshop Natural Language Processing for Social Media*, page 55, 2016.

[41] Hao Fang, Hao Cheng, and Mari Ostendorf. Learning latent local conversation modes for predicting community endorsement in online discussions. *arXiv preprint arXiv:1608.04808*, 2016.

[42] Atefeh Farzindar and Diana Inkpen. Natural language processing for social media. *Synthesis Lectures on Human Language Technologies*, 8(2):1–166, 2015.

[43] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A Kalyanpur, A. Lally, J W. Murdock, E. Nyberg, J. Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[44] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[45] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Aistats*, volume 15, page 275, 2011.

[46] Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, and Jaime Carbonell. Summarizing text documents: sentence selection and evaluation metrics. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 121–128. ACM, 1999.

[47] Yihong Gong and Xin Liu. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–25. ACM, 2001.

[48] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[49] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

[50] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *NIPS*, volume 1, pages 1523–1530, 2001.

[51] Hongyu Guo. Generating text with deep reinforcement learning. *arXiv preprint arXiv:1510.09202*, 2015.

[52] Qi Guo, Fernando Diaz, and Elad Yom-Tov. Updating users about time critical events. In Pavel Serdyukov, Pavel Braslavski, SergeiO. Kuznetsov, Jaap Kamps, Stefan Rger, Eugene Agichtein, Ilya Segalovich, and Emine Yilmaz, editors, *Advances in Information Retrieval*, volume 7814 of *Lecture Notes in Computer Science*, pages 483–494. Springer Berlin Heidelberg, 2013.

[53] Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard L Lewis, and Xiaoshi Wang. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *Advances in Neural Information Processing Systems*, pages 3338–3346, 2014.

[54] Sanda Harabagiu and Andrew Hickl. Relevance modeling for microblog summarization. In *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[55] Hado V. Hasselt. Double q-learning. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2613–2621. Curran Associates, Inc., 2010.

[56] M. Hausknecht and P. Stone. Deep reinforcement learning in parameterized action space. In *International Conference on Learning Representations*, 2016.

[57] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*, 2015.

[58] J. He, J. Chen, X. He, J. Gao, L. Li, L. Deng, and M. Ostendorf. Deep reinforcement learning with a natural language action space. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, 2016.

[59] J. He, M. Ostendorf, X. He, J. Chen, J. Gao, L. Li, and L. Deng. Deep reinforcement learning with a combinatorial action space for predicting popular reddit threads. In *Proc. of the 2016 Conference on Empirical Methods in Natural Language Processing*, November 2016.

[60] Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. Deep reinforcement learning with an action space defined by natural language. *arXiv preprint arXiv:1511.04636*, 2015.

[61] Ji He, Alex Marin, and Mari Ostendorf. Effective data-driven feature learning for detecting name errors in automatic speech recognition. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 230–235. IEEE, 2014.

[62] Ji He, Mari Ostendorf, and Xiaodong He. Reinforcement learning with external knowledge and two-stage q-functions for predicting popular reddit threads. *arXiv preprint arXiv:1704.06217*, 2017.

[63] Ji He, Mari Ostendorf, Xiaodong He, Jianshu Chen, Jianfeng Gao, Lihong Li, and Li Deng. Deep reinforcement learning with a combinatorial action space for predicting and tracking popular discussion threads. *arXiv preprint arXiv:1606.03667*, 2016.

[64] Ji He, Yao Qian, Frank K Soong, and Sheng Zhao. Turning a monolingual speaker into multilingual for a mixed-language tts. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[66] John R Hershey, Jonathan Le Roux, and Felix Weninger. Deep unfolding: Model-based inspiration of novel deep architectures. *arXiv preprint arXiv:1409.2574*, 2014.

[67] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.*, 29(6):82–97, 2012.

[68] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

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

[70] Liangjie Hong, Ovidiu Dan, and Brian Davison. Predicting popular messages in Twitter. In *Proc. Int. Conf. World Wide Web (WWW)*, pages 57–58, 2011.

[71] Chiao-Fang Hsu, Elham Khabiri, and James Caverlee. Ranking comments on the social web. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4, pages 90–97. IEEE, 2009.

[72] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[73] P-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proc. of the ACM International Conference on Information & Knowledge Management*, pages 2333–2338. ACM, 2013.

[74] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[75] A. Jaech, V. Zayats, H. Fang, M. Ostendorf, and H. Hajishirzi. Talking to the crowd: What do people react to in online discussions? In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 2026–2031, September 2015.

[76] Aaron Jaech, Victoria Zayats, Hao Fang, Mari Ostendorf, and Hannaneh Hajishirzi. Talking to the crowd: What do people react to in online discussions? *arXiv preprint arXiv:1507.02205*, 2015.

[77] C.D. Jones, A.B. Smith, and E.F. Roberts. Article title. In *Proceedings Title*, volume II, pages 803–806. IEEE, 2003.

[78] Karen Sparck Jones. What might be in a summary? *Information retrieval*, 93:9–26, 1993.

[79] B. Katz, G. Marton, G. C Borchardt, A. Brownell, S. Felshin, D. Loreto, J. Louis-Rosenberg, B. Lu, F. Mora, S. Stiller, et al. External knowledge sources for question answering. In *TREC*, 2005.

[80] Elham Khabiri, James Caverlee, and Chiao-Fang Hsu. Summarizing user-contributed comments. In *ICWSM*, 2011.

[81] Hyun Duk Kim and ChengXiang Zhai. Generating comparative summaries of contradictory opinions in text. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 385–394. ACM, 2009.

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

[83] R. Kiros, Y. Zhu, R. R Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3276–3284, 2015.

[84] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006.

[85] Andreas Krause and Carlos Guestrin. Near-optimal observation selection using submodular functions. In *AAAI*, volume 7, pages 1650–1654, 2007.

[86] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[87] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010.

[88] Himabindu Lakkaraju, Julian McAuley, and Jure Leskovec. What's in a name? Understanding the interplay between titles, content, and communities in social media. In *Proc. Int. AAAI Conf. Web and Social Media (ICWSM)*, 2013.

[89] C. Lampe and P. Resnick. Slash(dot) and burn: distributed moderation in a large online conversation space. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 543–550, 2004.

[90] Q. V Le and T. Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

[91] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[92] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*, 2016.

[93] J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas, November 2016. Association for Computational Linguistics.

[94] Jiwei Li, Sujian Li, Xun Wang, Ye Tian, and Baobao Chang. Update summarization using a multi-level hierarchical dirichlet process model. In *COLING*, pages 1603–1618, 2012.

[95] L. Li, W. Chu, J. Langford, and R. E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[96] L. Li, J. D. Williams, and S. Balakrishnan. Reinforcement learning for spoken dialog management using least-squares policy iteration and fast feature selection. In *Proceedings of the Tenth Annual Conference of the International Speech Communication Association (INTERSPEECH-09)*, page 24752478, 2009.

[97] X. Li, L. Li, J. Gao, X. He, J. Chen, L. Deng, and J. He. Recurrent Reinforcement Learning: A Hybrid Approach. *ArXiv e-prints*, September 2015.

[98] Xiujun Li, Lihong Li, Jianfeng Gao, Xiaodong He, Jianshu Chen, Li Deng, and Ji He. Recurrent reinforcement learning: A hybrid approach. *arXiv preprint arXiv:1509.03044*, 2015.

[99] T. P Lillicrap, J. J Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

[100] Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. Association for Computational Linguistics, 2003.

[101] J. J Lin. The web as a resource for question answering: Perspectives and challenges. In *LREC*, 2002.

[102] L-J Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3–4):293–321, 1992.

[103] L-J. Lin. Reinforcement learning for robots using neural networks. Technical report, DTIC Document, 1993.

[104] William A Little. The existence of persistent states in the brain. *Mathematical biosciences*, 19(1-2):101–120, 1974.

[105] Xiaohua Liu, Yitong Li, Furu Wei, and Ming Zhou. Graph-based multi-tweet summarization using social signals. In *COLING*, pages 1699–1714, 2012.

[106] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[107] Michal Lukasik, Trevor Cohn, and Kalina Bontcheva. Point process modelling of rumour dynamics in social media. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, 2015.

[108] M-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, September 2015.

[109] Alex Marin, Mari Ostendorf, and Ji He. Learning phrase patterns for asr name error detection using semantic similarity. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[110] M. Mathioudakis and N. Koudas. Twittermonitor: trend detection over the twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 1155–1158. ACM, 2010.

[111] Julie Medero and Mari Ostendorf. Identifying targets for syntactic simplification. In *SLaTE*, pages 69–72, 2011.

[112] Neville Mehta, Rakesh Gupta, Antoine Raux, Deepak Ramachandran, and Stefan Krawczyk. Probabilistic ontology trees for belief tracking in dialog systems. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 37–46. Association for Computational Linguistics, 2010.

[113] Clemens Meinhart. Studying user submissions and content on reddit.

[114] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[115] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

[116] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.

[117] A. Mnih and G. Hinton. Three new graphical models for statistical language modelling. In *Proc. of the 24th International Conference on Machine learning*, pages 641–648. ACM, 2007.

[118] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. *ArXiv e-prints*, December 2013.

[119] V. Mnih, K. Kavukcuoglu, D. Silver, A. A Rusu, J. Veness, M. G Bellemare, A. Graves, M. Riedmiller, A. K Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

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

[121] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[122] K. Narasimhan, T. Kulkarni, and R. Barzilay. Language understanding for text-based games using deep reinforcement learning. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, September 2015.

[123] K. Narasimhan, A. Yala, and R. Barzilay. Improving information extraction by acquiring external evidence with reinforcement learning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2355–2365, Austin, Texas, November 2016. Association for Computational Linguistics.

[124] Ani Nenkova and Kathleen McKeown. A survey of text summarization techniques. In *Mining Text Data*, pages 43–76. Springer, 2012.

[125] R. Nogueira and K. Cho. End-to-end goal-driven web navigation. In *Advances in Neural Information Processing Systems 29*, pages 1903–1911, 2016.

[126] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 1320–1326, 2010.

[127] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. Deep sentence embedding using the long short term memory network: Analysis and application to information retrieval. *arXiv preprint arXiv:1502.06922*, 2015.

[128] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.

[129] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[130] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

[131] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.

[132] Michael J Paul, ChengXiang Zhai, and Roxana Girju. Summarizing contrastive viewpoints in opinionated text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 66–76. Association for Computational Linguistics, 2010.

[133] J. Pennington, R. Socher, and C. D Manning. Glove: Global vectors for word representation. *Proc. of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543, 2014.

[134] Sasa Petrovic, Miles Osborne, and Victor Lavrenko. Rt to win! predicting message propagation in Twitter. In *Proc. Int. AAAI Conf. Web and Social Media (ICWSM)*, pages 586–589, 2011.

[135] Dragomir R Radev, Hongyan Jing, and Malgorzata Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic summarization*, pages 21–30. Association for Computational Linguistics, 2000.

[136] Predicting responses to Microblog posts. Predicting responses to microblog posts. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 602–606, 2012.

[137] B. Richard. *Dynamic programming*. Princeton University Press, 1957.

[138] Andrew; Ernest Adams (2006). Fundamentals of Game Design. Prentice Hall. Rollings. *Rollings, Andrew; Ernest Adams (2006). Fundamentals of Game Design. Prentice Hall.*

[139] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[140] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[141] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[142] B. Sallans and G. E Hinton. Reinforcement learning with factored states and actions. *The Journal of Machine Learning Research*, 5:1063–1088, 2004.

[143] G. Salton and M. J McGill. *Introduction to modern information retrieval*. McGraw-Hill, Inc., 1986.

[144] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

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

[146] K. Scheffler and S. Young. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proc. of the second International Conference on Human Language Technology Research*, pages 12–19, 2002.

[147] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[148] Holger Schwenk. Continuous space language models. *Computer Speech & Language*, 21(3):492–518, 2007.

[149] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM, 2014.

[150] D. Silver, A. Huang, C. J Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[151] S. P Singh, M. J Kearns, D. J Litman, and M. A Walker. Reinforcement learning for spoken dialogue systems. In *NIPS*, pages 956–962, 1999.

[152] Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, pages 105–133, 2002.

[153] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[154] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan. A neural network approach to context-sensitive generation of conversational responses. In *NAACL-HLT 2015*, 2015.

[155] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[156] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[157] Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689*, 2016.

[158] B. Suh, L. Hong, P. Pirolli, and E. H. Chi. Want to be retweeted? Large scale analytics on factors impacting retweet in twitter network. In *Proc. IEEE Inter. Conf. on Social Computing (SocialCom)*, pages 177–184, 2010.

[159] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[160] Tao Sun, Ming Zhang, and Qiaozhu Mei. Unexpected relevance: an empirical study of serendipity in retweets. In *Proc. Int. AAAI Conf. Weblogs and Social Media (ICWSM)*, pages 592–601, 2013.

[161] I. Sutskever, O. Vinyals, and Q. V Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[162] Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton. On the importance of initialization and momentum in deep learning. *ICML (3)*, 28:1139–1147, 2013.

[163] R. S Sutton and A. G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[164] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063, 1999.

[165] Chenhao Tan, Lillian Lee, and Bo Pang. The effect of wording on message propagation: Topic- and author-controlled natural experiments on Twitter. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 175–186, 2014.

[166] Chenhao Tan, Lillian Lee, and Bo Pang. The effect of wording on message propagation: Topic-and author-controlled natural experiments on twitter. *arXiv preprint arXiv:1405.1438*, 2014.

[167] Manos Tasgkias, Wouter Weerkamp, and Maarten de Rijke. Predicting the volume of comments on online news stories. In *Proc. CIKM*, pages 1765–1768, 2009.

[168] Alexandru Tatar, Jeremie Leguay, Panayotis Antoniadis, Arnaud Limbourg, Marcelo Dias de Amorim, and Serge Fdida. Predicting the polularity of online articles based on user comments. In *Proc. Inter. Conf. on Web Intelligence, Mining and Semantics (WIMS)*, pages 67:1–67:8, 2011.

[169] G. Tesauro. Temporal difference learning and TD-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[170] 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(2), 2012.

[171] Ivan Titov and Ryan T McDonald. A joint model of text and aspect ratings for sentiment summarization. In *ACL*, volume 8, pages 308–316. Citeseer, 2008.

[172] John N Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. *Automatic Control, IEEE Transactions on*, 42(5):674–690, 1997.

[173] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. *CoRR, abs/1509.06461*, 2015.

[174] O. Vinyals, Ł. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781, 2015.

[175] Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. *arXiv preprint arXiv:1412.7449*, 2014.

[176] Marilyn A. Walker. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*, pages 387–416, 2000.

[177] Zhuoran Wang and Oliver Lemon. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the SIGDIAL 2013 Conference*, pages 423–432, 2013.

[178] Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[179] C. JCH Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[180] T.-H. Wen, M. Gasic, N. Mrksic, L. M Rojas-Barahona, P.-H. Su, S. Ultes, D. Vandyke, and S. Young. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*, 2016.

[181] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*, 2015.

[182] T. Weninger, X. A. Zhu, and J. Han. An exploration of discussion threads in social news sites: A case study of the reddit community. In *Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on*, pages 579–583. IEEE, 2013.

[183] Jason D Williams. The best of both worlds: unifying conventional dialog systems and pomdps. In *INTERSPEECH*, pages 1173–1176, 2008.

[184] Jason D Williams and Steve Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007.

[185] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, pages 229–256, 1992.

[186] F. Wu and D. 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.

[187] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[188] Rui Yan, Mirella Lapata, and Xiaoming Li. Tweet recommendation with graph co-ranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 516–525. Association for Computational Linguistics, 2012.

[189] Z. Yan, N. Duan, J. Bao, P. Chen, M. Zhou, Z. Li, and J. Zhou. Docchat: An information retrieval approach for chatbot engines using unstructured documents. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 516–525, Berlin, Germany, August 2016. Association for Computational Linguistics.

[190] H. Yang, T-S Chua, S. Wang, and C-K Koh. Structured use of external knowledge for event-based open domain question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 33–40. ACM, 2003.

[191] Tae Yano and Noah A. Smith. What's worthy of comment? Content and comment volume in political blogs. In *Proc. Int. AAAI Conf. Weblogs and Social Media (ICWSM)*, 2010.

[192] Steve Young, Milica Gasic, Blaise Thomson, and John D Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.

[193] Y. Yue and C. Guestrin. Linear submodular bandits and their application to diversified retrieval. In *Advances in Neural Information Processing Systems*, pages 2483–2491, 2011.

[194] Vicky Zayats and Mari Ostendorf. Conversation modeling on reddit using a graph-structured lstm. *arXiv preprint arXiv:1704.02080*, 2017.

[195] Qingyuan Zhao, Murat A. Erdogdu, Hera Y. He, Anand Rajaraman, and Jure Leskovec. SEISMIC: A self-exciting point process model for predicting Tweet popularity. In *Proc. ACM SIGKDD Conf. Knowledge Discovery and Data Mining*, 2015.

[196] Arkaitz Zubiaga, Damiano Spina, Enrique Amigó, and Julio Gonzalo. Towards real-time summarization of scheduled events from twitter streams. In *Proceedings of the 23rd ACM conference on Hypertext and social media*, pages 319–320. ACM, 2012.