DEEP REINFORCEMENT LEARNING WITH AN UNBOUNDED ACTION SPACE

Ji He[*], Jianshu Chen[†], Xiaodong He[†], Jianfeng Gao[†], Lihong Li[†], Li Deng[†], Mari Ostendorf[*]
[*]Department of Electrical Engineering
University of Washington
Seattle, WA 98195, USA
{jvking, ostendor}@uw.edu

[†]Microsoft Research
Redmond, WA 98052, USA
{jianshuc, xiaohe, jfgao, lihongli, deng}@microsoft.edu

ABSTRACT

In this paper, we propose the deep reinforcement relevance network (DRRN), a novel deep architecture, to design a better model for handling an unbounded action space with applications to language understanding for text-based games. For a particular class of games, a user must choose among a variable number of actions described by text, with the goal of maximizing long-term reward. In these games, the best action is typically that which best fits to the current situation (modeled as a state in the DRRN), also described by text. Because of the exponential complexity of natural language with respect to sentence length, there is typically an unbounded set of unique actions. Therefore, it is difficult to pre-define the action set. To address this challenge, the DRRN extracts separate high-level embedding vectors from the texts that describe states and actions, respectively, using a general interaction function, exploring inner product, bilinear, and DNN interaction, between these embedding vectors to approximate the Q-function. We evaluate the DRRN on two popular text games, showing superior performance over other deep Q-learning architectures.

1 INTRODUCTION

We consider a sequential text understanding and decision making problem with an unbounded action space in the context of learning to play text games. At each time step, the learning agent is given a text string that describes a certain state of the game (i.e., environment, defined as the "state text") and several text strings that describe all the possible actions one could take (defined as the "action texts"). After selecting one of the actions (e.g., by clicking the corresponding hyperlink of that text), the environment will move to another state and further reveal some texts about the next state. The reward signal could either be given during each transition or in the end. The objective is to understand, at each step, the state text and all the action texts to pick up the best action so that we *navigate through the sequence of texts with the highest long-term reward* (e.g., the game reaches a good ending of the story). To this end, the learning agent needs to find the most relevant action text for the current state text. Here we define a new notion of relevance based on their joint impact on the reward: an action text string is said to be "more relevant" (to a state text string) than the other action texts if taking that action would lead to higher long-term reward. We formulate the problem as a reinforcement learning problem and propose a novel deep reinforcement relevance network (DRRN) for the text understanding and navigation task. Deep neural networks (DNN) are used to map text strings into embedding vectors in a common finite-dimensional space, where "relevance" is measured numerically by a general interaction function, such as their inner product. In particular, the outcome of the inner product defines the value of the Q-function for the current state-action pair, which characterizes the long-term reward for pairing these two text strings. The embedding and the Q-function will be learned in an end-to-end manner by maximizing the long-term reward.

This work is a novel contribution to deep reinforcement learning in terms of the representation of text-based actions and states. Recently, deep reinforcement learning using a deep Q-network (DQN) has been shown to outperform human level performance in several benchmarks of Atari games (Mnih et al., 2015). In the DQN, a deep convolutional neural network is used to extract high-level features from images, which are further mapped into Q-function values of different actions via a linear transform. In Atari games, the action sets are pre-defined and fixed during the entire learning and decision making process. However, in text understanding, this is usually not the case since each action in itself might be a string of text. Because of exponential complexity of natural language with respect to sentence length, there could be unboundedly many unique actions in this scenario. Therefore, in order to understand texts and make the best decision, we need to extract fixed-dimension features (embedding vectors) not only from the text that describes the state, but also from the text that describes each action. This is an important structural difference from the prior art of DQN, which only extracts a fixed-dimension embedding from the state side. Enabled by this novel structure in the DRRN, the number of actions can be different at different steps of the decision making process. In this paper, we will evaluate our proposed method on two text games, which show its superior performance over the previously published DQN architectures.

The rest of the paper is organized as follows. In Section 2, we review related works. Then, we develop the deep reinforcement relevance network in Section 3, which is evaluated on two text games in Section 4. Finally, in Section 5, we conclude with suggestions for future work.

2 Related Work

Reinforcement learning is an approach to making decisions that maximizes long-term rewards (Sutton & Barto, 1998). One of the first successful reinforcement learning applications using neural network function approximation is TD-gammon (Tesauro, 1995). Recently, inspired by advances in deep learning (LeCun et al., 2015; Hinton et al., 2012; Krizhevsky et al., 2012; Dahl et al., 2012), significant progress has been made by combining deep learning with reinforcement learning. The "Deep Q-Network" (DQN) was developed and applied to Atari games (Mnih et al., 2013; Mnih et al., 2015) and was shown to achieve human level performance by applying convolutional neural networks to the raw image pixels. A deep neural network is used as a function approximation in a variant of Q-learning (Watkins & Dayan, 1992), and a couple of techniques are introduced to ensure the algorithm converges stably. Another stream of work focuses on continuous control with deep reinforcement learning (Lillicrap et al., 2015), where an actor-critic algorithm operates over a continuous action space. A continuous action space differs from an unbounded action space in that the action space is known. The DRRN framework treats text games as a black-box; thus, the potential action space defined through natural language is unbounded.

There has also been increasing interest in applying reinforcement learning, especially DQN, to other problems. Li et al. (2015) developed a joint training approach for recurrent reinforcement learning and demonstrate its effectiveness on a customer relationship management task. 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 (Scheffler & Young, 2002; Singh et al., 1999). There has also been interest in extracting textual knowledge to improve game control performance (Branavan et al., 2011), and mapping natural language instructions to sequences of executable actions (Branavan et al., 2009). Narasimhan et al. (2015) 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 (Narasimhan et al., 2015), such as fixed command structures (one action and one argument object), and limited action-side vocabulary size. For natural language, this approach has the drawback of a complex action space, and it is infeasible for Choice-based or Hypertext text games (see a brief review of different text games in Section 4.1). To the best of our knowledge, deep reinforcement learning for text games with variable natural language actions is relatively limited.

Another stream of relevant work is text embedding. Bengio et al. (2003) introduced a neural probabilistic language model that learns a distributed representation of words. Mnih & Hinton (2007) proposed three graphical models for statistical language modeling and showed the advantage of using a log-bilinear language model. Collobert & Weston (2008) described a convolutional neural

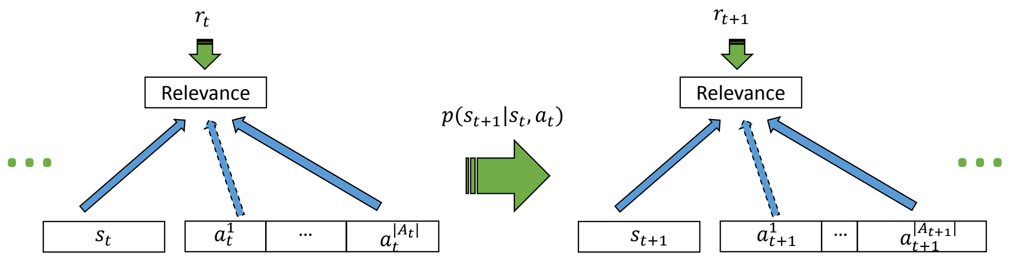


Figure 1: The Markov decision process for text understanding. The green arrows represent the transition of the MDP and the signals revealed by the MDP to the agent. The blue arrows denote the information processing done by the agent. Each state s_t is described by a certain text, and each action a_t is also described by another text. Note that the number of actions, $|\mathcal{A}_t|$, depends on t, and thus could vary over time. Given the state-text, the agent learns to choose the best action-text to maximize the long-term rewards.

network architecture that is trained jointly with multitask learning. Mikolov et al. (2013) introduced word2vec, which is an efficient estimation of continuous vector representations of words. They further explored distributed representations of sentences and documents (Le & Mikolov, 2014) and showed that variable-length pieces of texts can be represented by a dense fixed-length vector. Pennington et al. (2014) presented a log-bilinear model that combines the advantages of global matrix factorization and local context window methods. Kiros et al. (2015) described an approach for unsupervised learning of reconstructing the surrounding sentences of an encoded passage using an encoder-decoder model. In web search areas, Huang et al. (2013) developed the Deep Structured Semantic Model that uses deep neural networks to approximate this projection at a semantic level, and measure relevance by computing cosine similarity.

3 Deep Reinforcement Relevance Network

3.1 Sequential decision making in text understanding

We consider the sequential decision making problem for text understanding in Figure 1, which we model as a Markov decision process (MDP). At each time step t, the agent will receive a string of text that describes the state s_t (i.e., "state-text") and several strings of text that describe all the potential actions a_t (i.e., "action-text"). The agent is required to understand the texts from both the state side and the action side, measuring their relevance to the current context s_t for the purpose of maximizing the long-term reward, and then picking up the best action a_t^*. Then, the environment will transition to another state $s_{t+1} = s'$ according to the probability $p(s'|s, a)$, and the agent will receive a reward r_t for that particular transition. The *policy* of the agent is defined to be the probability $\pi(a_t|s_t)$ of taking action a_t at state s_t. We further define the Q-function $Q(s, a)$ that characterizes the expected return starting from s, taking the action a, and thereafter following policy $\pi(a|s)$ to be:

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

where γ denotes a discount factor. The optimal $Q^*(s, a)$ is shown to be the solution to the Bellman Equation (Richard, 1957; Sutton & Barto, 1998):

$$Q^*(s,a) = \mathbb{E}\big[r_t + \gamma \max_a Q^*(s', a) \big| s_t = s, a_t = a\big] \tag{2}$$

where the expectation is evaluated with respect to the randomness of $s_{t+1} = s'$ and r_t given $s_t = s$ and $a_t = a$. The optimal policy and Q-function could be found by using the Q-learning algorithm (Watkins & Dayan, 1992):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta_t \cdot \big(r_t + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\big) \tag{3}$$

where η_t is the learning rate of the algorithm. In this paper, we use a softmax selection strategy as the exploration policy during the learning stage, which chooses the action a_t at state s_t according to the following probability:

$$\pi(a_t = a_t^i | s_t) = \frac{\exp(\alpha \cdot Q(s_t, a_t^i))}{\sum_{j=1}^{|\mathcal{A}_t|} \exp(\alpha \cdot Q(s_t, a_t^j))}, \quad i = 1, \ldots, |\mathcal{A}_t| \tag{4}$$

where $\mathcal{A}_t$ denotes the set that contains all the feasible actions at state s_t, a_t^i denotes the i-th feasible actions in $\mathcal{A}_t$, $| \cdot |$ denotes the cardinality of the set, and α denotes the scaling factor in the softmax operation.[1] We choose this to be a constant throughout the learning period. Since all methods in this paper initialize with small random weights, initial Q-value differences will be small, thus making the Q-learning more explorative at the beginning. As Q-values better approximate the true values, a reasonable α will make action selection put high probability on the optimal action (exploitation), but still maintain a small explorative probability.

3.2 THE UNBOUNDED ACTION SPACE

Let $\mathcal{S}$ denote the state space, and let $\mathcal{A}$ denote the entire action space that includes all the unique actions over time. A vanilla Q-learning recursion (3), which needs to maintain a table of size $|\mathcal{S}| \times |\mathcal{A}|$, is not applicable to a large state space problem. The prior work of DQN focused on using a DNN in Q-function approximation and has shown high capacity and scalability, leading to state-of-the-art performance in many machine learning tasks. Specifically, the DQN takes the raw data of the state (e.g., the text or the image pixels) as its input and generates $|\mathcal{A}|$ outputs, each of which represents the value of $Q(s, a)$ for a particular action a. This architecture has been successfully applied to solve Atari games, where the action space is pre-defined and fixed to be a small set over time.

However, the standard DQN could not handle the unbounded action space $\mathcal{A}$. Indeed, the feasible action set $\mathcal{A}_t$ at each time t could be an unknown subset of the unbounded action space $\mathcal{A}$, and could be different over time (see Figure 1), which is a common situation that arises in Choice-based and Hypertext-based text games. In Figure 2, a modified DQN is illustrated, which concatenates the state and action vectors (i.e., the extended state vector) as its input and computes the Q-function values for the actions in the current feasible action set $\mathcal{A}_t$ as its outputs. This architecture requires the model to know the maximum number of feasible actions (i.e., $\max_t |\mathcal{A}_t|$) as a priori. However, it is important to note that the value of $\max_t |\mathcal{A}_t|$ is difficult to obtain in practice because $\mathcal{A}_t$ is usually unknown beforehand. Nevertheless, we will still use this modified DQN (with the knowledge of $\max_t |\mathcal{A}_t|$) as a baseline for comparison to our newly developed DRRN architecture in the next subsection, which does not require the knowledge of $\max_t |\mathcal{A}_t|$, while still outperforming DQN.

Apart from the modified DQN, other baselines could be used. For example, function approximation using a neural network (NN-RL) that takes a state-action pair as input, and output a single Q-value. In fact, this architecture can handle a varying number of feasible actions. While this might be a more general way of correlating state-action pair, it does not learn text embeddings from raw text input, and thus may not fully incorporate domain knowledge or language understanding. The main advantage of having a distributed representation of text is that it can capture various dimensions of both semantic and syntactic information in a vector. As a result, the text embedding is compact and less susceptible to data sparsity. 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 individual deep neural networks for state and action sides, we can extract text embeddings and further apply various interaction functions.

3.3 DRRN ARCHITECTURE: FORWARD ACTIVATION

Motivated by the aforementioned challenges, we develop deep reinforcement relevance network (DRRN), which is a novel deep architecture for handling an unbounded action space in sequential text understanding. As shown in Figure 3, the DRRN consists of a pair of DNNs, one for state text embedding and the other for action text embedding. Given any state/action text pair (s_t, a_t^i), the

[1] α controls the balance between exploitation and exploration. For instance, if α = 0, the actions are uniformly sampled, whereas $\alpha = \infty$, there is no exploration.

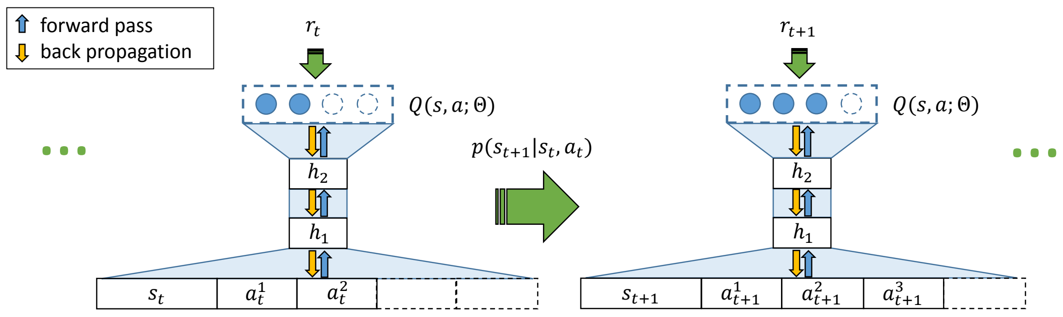


Figure 2: A modified DQN architecture. In a traditional DQN, only the state text is embedded into a vector space, which is further mapped into a fixed set of outputs. Action texts in choice-based/hypertext text games are converted to feature representations and concatenated with the state representation as the DQN input. As there are a variable number of actions ($|\mathcal{A}_t| = 2$ and $|\mathcal{A}_{t+1}| = 3$ as shown in the figure), a maximum number of action slots needed throughout the game are pre-allocated for the input vector (in this case 4). The output layer of the network also has 4 nodes. However at time t, the DQN only considers $|\mathcal{A}_t|$ nodes (corresponding to the input actions).

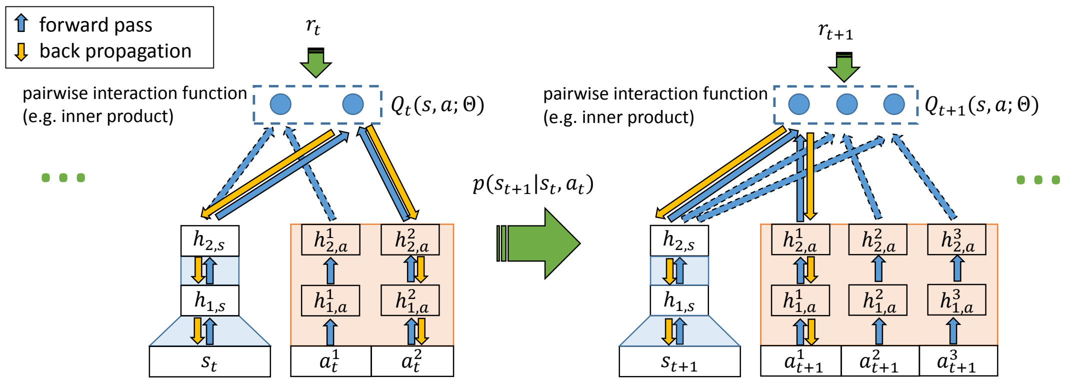


Figure 3: The DRRN architecture. There are a total of $|\mathcal{A}_t|$ deep networks for the action side, one for each action text. The orange box around these $|\mathcal{A}_t|$ networks implies that their model parameters are shared across each other. This special structure endows the model the ability to handle a varying number of actions ($|\mathcal{A}_t| = 2$ and $|\mathcal{A}_{t+1}| = 3$ as shown in the figure) across different steps.

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 general interaction functions 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 $a_t = \arg\max_{a_t^i} Q(s_t, a_t^i)$.

While we allow the DNN at the action side to have different model parameters from the DNN on the state side, the DNNs for the action-texts are set to have *tied* model parameters. The reason for having tied DNN model parameters on the action side is that the texts that describe different actions usually share common statistics and it is thus natural to use a common model to extract the corresponding high level representations from them. This tied structure enables each training sample to be shared with the DNNs on all the actions, making them more efficiently trained by the existing data samples. This is in contrast to the DQN structure (see Figure 2), where different texts for different actions are assigned independent model parameters, which does not make full use of the particular structure inherent in this text understanding problem. On the other hand, the text that describes the state could have different functions from the texts on the action side. Therefore, the parameters of the DNN on the state side are learned separately. 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

Algorithm 1 Learning algorithm for DRRN

```
1: Initialize replay memory $\mathcal{D}$ to capacity $N$.
2: Initialize DRRN with small random weights.
3: Initialize game simulator and load dictionary.
4: for $episode = 1, \ldots, M$ do
5:     Restart game simulator.
6:     Read raw state text and a list of action text from the simulator, and convert them to representation $s_1$ and $a_1^1, a_1^2, \ldots, a_1^{|\mathcal{A}_1|}$.
7:     for $t = 1, \ldots, T$ do
8:         Compute $Q(s_t, a_t^i; \Theta)$ for the list of actions using DRRN forward activation (Section 3.3).
9:         Select an action $a_t$ based on probability distribution $\pi(a_t = a_t^i | s_t)$ (Equation 4)
10:        Execute action $a_t$ in simulator
11:        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 $a_{t+1}^1, a_{t+1}^2, \ldots, a_{t+1}^{|\mathcal{A}_{t+1}|}$.
12:        Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$.
13:        Sample random mini batch of transitions $(s_k, a_k, r_k, s_{k+1})$ from $\mathcal{D}$.
14:        Set $y_k = \begin{cases} r_k & \text{if } s_{k+1} \text{ is terminal} \\ r_k + \gamma \max_{a'} Q(s_{k+1}, a'; \Theta)) & \text{otherwise} \end{cases}$
15:        Perform a gradient descent step on $(y_k - Q(s_k, a_k; \Theta))^2$ with respect to the network parameters $\Theta$ (Section 3.4). Back-propagation is performed only for $a_k$ even though there are $|\mathcal{A}_k|$ actions at time $k$.
16:    end for
17: end for
```

linear transformation weight matrix and bias vector between the $(l-1)$-th and l-th hidden layers, respectively. $W_{l,a}$ and $b_{l,a}$ denote the equivalent parameters for the action side. In this study, the DRRN has L hidden layers on each side.

$$h_{1,s} = f(W_{1,s} s_t + b_{1,s}) \tag{5}$$

$$h_{1,a}^i = f(W_{1,a} a_t^i + b_{1,a}), \quad i = 1, 2, 3, ..., |\mathcal{A}_t| \tag{6}$$

$$h_{l,s} = f(W_{l-1,s} h_{l-1,s} + b_{l-1,s}), \quad l = 2, 3, ..., L \tag{7}$$

$$h_{l,a}^i = f(W_{l-1,a} h_{l-1,a}^i + b_{l-1,a}), \quad i = 1, 2, 3, ..., |\mathcal{A}_t|, l = 2, 3, ..., L \tag{8}$$

where $f(\cdot)$ is the nonlinear activation function at the hidden layers, which, for example, could be chosen as $\tanh(x)$, and $\mathcal{A}_t$ denotes the set of feasible actions at time t. A general interaction function $g(\cdot)$ is used to approximate the Q-function values, $Q(s, a)$, in the following parametric form:

$$Q(s_t, a_t^i; \Theta) = g\left(h_{L,s},\ h_{L,a}^i\right) \tag{9}$$

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.

The success of the DRRN in handling unbounded $\mathcal{A}$ lies in the fact that both the state-text and the action-texts are mapped into a finite-dimensional embedding space and the fact that DNNs on the action sides have tied parameters. The end-to-end learning process (discussed in the next subsection) will make the embedding vector of the state-text more aligned with the embedding vector of the "good" (or relevant) action-text than the "bad" (or irrelevant) ones so that their corresponding inner product, which is the Q-function value of taking this action, is higher.

3.4 Learning the DRRN: Back propagation

To learn the DRRN, we use the "experience-replay" strategy (Lin, 1993), which uses a fixed exploration policy to interact with the environment to obtain a data trajectory. Then, we randomly sample a transition tuple (s_k, a_k, r_k, s_{k+1}), compute the temporal difference error $d_k = r_k + \gamma \cdot \max_a Q(s_{k+1}, a; \Theta_{k-1}) - Q(s_k, a_k; \Theta_{k-1})$, and update the model according to the follow-

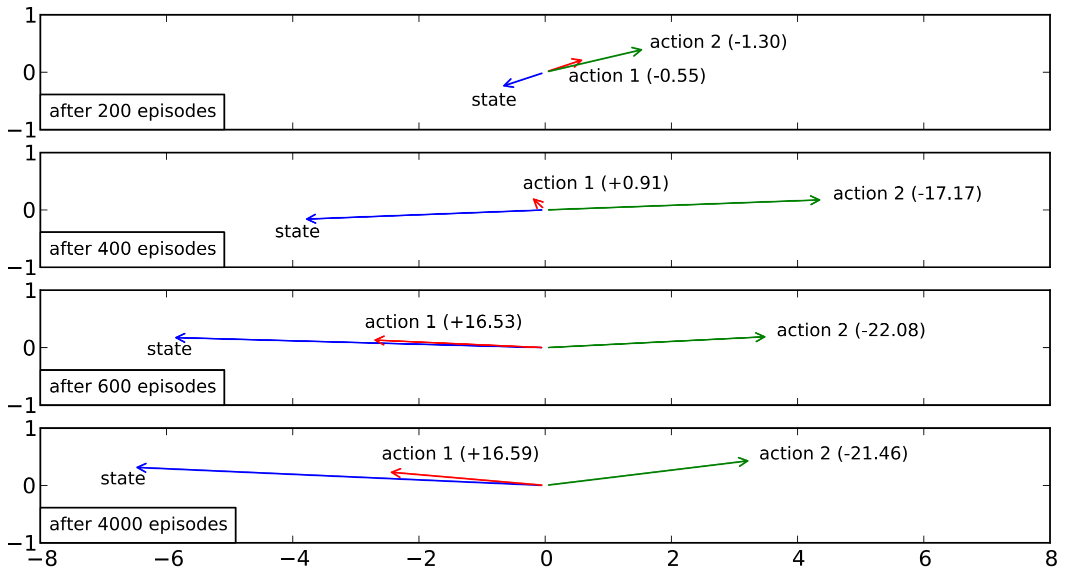


Figure 4: During DRRN model training, use PCA to project each text embedding vector to a 2-D plane. For each action, Q-values are in parentheses. In this example, state is "As you move forward, the people surrounding you suddenly look up with terror in their faces, and flee the street.". Action 1 (the positive action) is "Look up.", and Action 2 (the negative action) is "Ignore the alarm of others and continue moving forward."

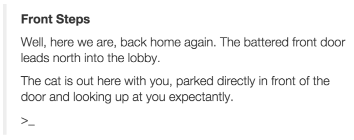
ing recursions:

$$W_{s,k} = W_{s,k-1} + \eta_k d_k \cdot \frac{\partial Q(s_k, a_k; \Theta_{k-1})}{\partial W_s}, \quad b_{s,k} = b_{s,k-1} + \eta_k d_k \cdot \frac{\partial Q(s_k, a_k; \Theta_{k-1})}{\partial b_s} \tag{10}$$

$$W_{a,k} = W_{a,k-1} + \eta_k d_k \cdot \frac{\partial Q(s_k, a_k; \Theta_{k-1})}{\partial W_a}, \quad b_{a,k} = b_{a,k-1} + \eta_k d_k \cdot \frac{\partial Q(s_k, a_k; \Theta_{k-1})}{\partial b_a} \tag{11}$$

where the expressions for $\frac{\partial Q}{\partial W_s}$, $\frac{\partial Q}{\partial W_a}$, $\frac{\partial Q}{\partial b_s}$, $\frac{\partial Q}{\partial b_a}$ are given in Appendix B. The notation k denotes the index of the transition tuple sampled from the trajectory. This 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 (Mnih et al., 2015). As illustrated in Figure 3, since the models on the action side are tied, models associated with other actions will also be updated even though the back propagation is only over one action.

The full algorithm is summarized in Algorithm 1. Note from Figure 3 that 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 will be *automatically learned* to be aligned with each other in the text embedding space from the reward signals. In Figure 4, we used Principal Component Analysis (PCA) to project the 100-dimension last hidden layer representation (before inner product) to a 2-D plane. The vector embeddings start with small values, due to small random initialization in model parameters, and after 600 episodes of experience replay training, the text embedding vectors are close to the converged embedding vectors (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.



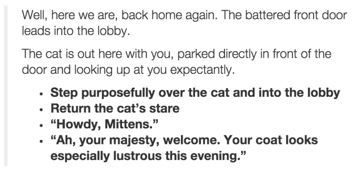
Front Steps

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.

>_

(a) Parser-based

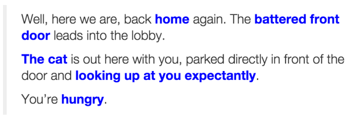


Well, here we are, back home again. The battered front door leads into the lobby.

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."**
- **"Ah, your majesty, welcome. Your coat looks especially lustrous this evening."**

(b) 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**.

(c) Hypertext-based

Figure 5: Different types of text games[4].

4 Experimental Results

4.1 Overview of text games

Text games, although simple compared to video games, still enjoy high popularity in online communities,[2] with annual competitions[3] held online since 1995. Text games communicate to players in the form of text display, which players have to understand and respond to by typing or clicking text (Adams, 2014).

There are three different types of text games: parser-based (Figure 5(a)), choice-based (Figure 5(b)), and hypertext-based (Figure 5(c)). Parser-based games were popular among early personal computer users, and 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, do not require a text parser. These two types of text games present actions 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 percentage from 8% in 2010 to 62% in 2014.[5]

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, reinforcement learning is appropriate for modeling long-term dependency in text games.

Compared to parser-based text games, choice-based and hypertext-based text games are especially difficult to solve automatically. This is due to the potentially unbounded representation of natural language sentences. Thus they serve as good tasks for the DRRN. Previously for parser-based text games, Narasimhan et al. (2015) have defined a fixed set of 222 actions, which is the total number of possible phrases the parser accepts. Parser-based text games are reduced to a problem that is solvable by a standard DQN. However, for choice-based and hypertext-based text games, the size of the action space is exponential with the length of the action sentences. Furthermore, another salient feature of choice-based and hypertext-based text games is that they usually have a variable number of input actions, which can be difficult to integrate into a fixed-output neural network architecture like a DQN. We will show that the new DRRN model is better suited because it solves both the problems of an unbounded action space and variable input actions, by properly combining the text understanding power of DNN with reinforcement learning in a new deep learning architecture.

In this study, we evaluate the DRRN with two games: a deterministic text game task called "Saving John" and a larger-scale stochastic text game called "Machine of Death" from a public archive.[6] The

[2]http://www.intfiction.org/forum/, http://ifdb.tads.org/
[3]http://www.ifcomp.org/
[4]http://www.ifcomp.org/about/if
[5]Statistics are obtained from http://www.ifarchive.org
[6]http://www.ifarchive.org

Stats	"Saving John"	"Machine of Death"
Text game type	Choice-based	Choice-based & Hypertext-based
Vocab size	1762	2258
Action vocab size	171	419
Avg. words/description	76.67	67.80
State transitions	Deterministic	Stochastic
# of states (underlying)	≥ 70	≥ 200
(Avg., max) steps/episode	$14, \geq 38$	$83, \geq 500$

Table 1: Comparison on text statistics for the games "Saving John" and and "Machine of Death".

basic text statistics of these tasks are shown in Table 1. The maximum value of feasible actions (i.e., $\max_t |\mathcal{A}_t|$) is four in "Saving John", and nine in "Machine of Death".

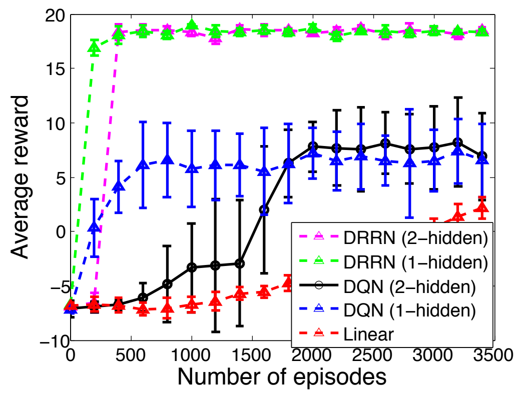
4.2 Experiment setup

The DRRN is evaluated on the two games described above. We manually annotate final rewards for all distinct endings in both games (as shown in Appendix C). The magnitude of reward scores are given to describe sentiment polarity of good/bad endings. On the other hand, each non-terminating step we assign with a small negative reward. For the text game "Machine of Death", we restrict an episode to be no longer than 500 steps. 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. In our examples, we use different vocabularies for the state side and action side. For text input, we use raw bag-of-words as features. The deterministic nature of state transitions in "Saving John" might lead to faster convergence than stochastic games, but as we will see in next paragraphs, the exploration-exploitation in learning will always add stochasticity to training.

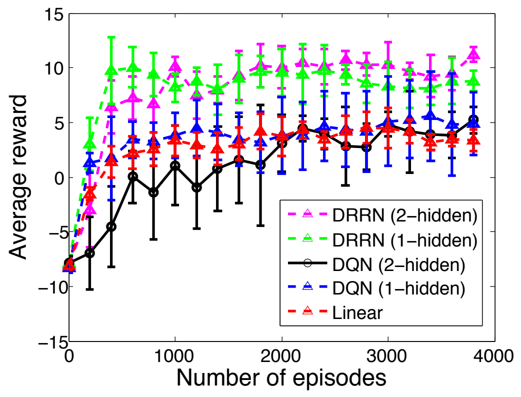
We apply DRRN with both 1 hidden layer and 2 hidden layer structures. For simplicity, we set the hidden dimension to be the same for each hidden layer, and for both state side and action side. In this way we could directly compute the inner product at the last hidden layer to approximate Q-function values. We use DRRNs with 100-dimension hidden layer(s) and build learning curves during experience replay training. In testing, we apply softmax selection. This is consistent with training, and the resulting average rewards will not converge to one particular ending. We record average final rewards as performance of the model. The learning rate η_t is set to be 0.001 throughout time.

The DRRN is compared to multiple baselines: a linear model, two DQNs (with one hidden layer and two hidden layers, respectively) and a NN-RL (with two hidden layers). 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 DQN baselines, the input is the text strings of state and action descriptions together as a bag of words, with the number of outputs equal to the maximum number of actions. When there are fewer actions than the maximum, then the highest scoring available action is used. The NN-RL baseline takes each pair of state-action texts as input, and generates a corresponding Q-value.

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 in (4), where we set $\alpha = 0.2$ for the first game and $\alpha = 1.0$ for the second game. We then shuffle the generated data tuples (s_t, a_t, r_t, s_{t+1}) and apply Algorithm 1 to update the model. 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



(a) Game 1: "Saving John"



(b) Game 2: "Machine of Death"

Figure 6: Learning curves of the two text games.

Evaluation metric	Average reward			
hidden dimension	10	20	50	100
Linear	4.46 (0.48)			
DQN (1-hidden)	2.68 (2.22)	2.92 (3.14)	4.05 (4.21)	5.98 (2.57)
DQN (2-hidden)	-3.46 (2.20)	4.90 (3.24)	9.01 (3.25)	7.12 (3.18)
DRRN (1-hidden)	11.47 (1.10)	17.14 (0.62)	18.33 (0.28)	18.22 (0.22)
DRRN (2-hidden)	13.84 (0.72)	18.45 (0.11)	18.51 (0.35)	**18.70** (0.41)

Table 2: The final average rewards and standard deviations on "Saving John".

randomly shuffle the list of actions.[7] This will enforce the algorithms to make decisions based on the understanding of the texts that describe the states and actions.

4.3 Performance

In Figure 6, we show the learning curves of different models, where the dimension of the hidden layers in DQN and DRRN are all set to 100. The error bars are obtained by running five independent experiments. As we see from Figure 6, both proposed methods and baseline methods started at about the same performance (-6.9 $\sim$ -7.1 average rewards for Game 1, and -7.9 $\sim$ -8.2 average rewards for Game 2; most of the time falling into bad endings), which is the random guess policy. After around 3000 episodes of experience replay training, all methods converge. The DRRN converges much faster than the other three baselines and achieves a better long-term reward. We hypothesize this is because the DRRN architecture is better at capturing relevance between state text and action text. For the linear and DQN methods, concatenating input features does not make effective utilization of information from both sides. In fact, the DRRN contains far fewer free parameters compared to the DQN (in "Machine of Death", DRRN with 2 hidden layers has 288k free parameters, compared to 614k free parameters in DQN with 2 hidden layers). We also test and find that shared embeddings between the state and action nets would speed up learning. However, the learning curve is less stable and the final converged performance is slightly worse (10.5 vs. 11.2), as shown in Appendix D.

The final performance (at convergence) for both baselines and proposed methods are shown in Tables 2 and 3. We test for different model sizes with 10, 20, 50, and 100 dimensions in the hidden layers. We see that the DRRN performs consistently better than all baselines (including DQN), and often with a lower variance.[8] The converged performance continues to grow by increasing the number

[7]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 person is in this same state. We want to force the models (both baselines and proposed) to learn the relevance between the words associated with a state and actions rather than memorizing a particular action presentation order.

[8]For Game 2 scores, random policy achieves -6.95. Due to the complexity of this game's underlying state transition, we cannot compute the exact optimal policy score, but the average optimal policy score should be

Evaluation metric	Average reward			
hidden dimension	10	20	50	100
Linear	3.38 (1.02)			
DQN (1-hidden)	2.11 (1.60)	2.05 (1.26)	3.76 (1.63)	4.89 (2.91)
DQN (2-hidden)	-0.43 (3.32)	2.82 (0.98)	4.35 (0.95)	5.23 (1.20)
DRRN (1-hidden)	7.69 (1.26)	7.24 (1.59)	8.47 (1.36)	8.79 (0.98)
DRRN (2-hidden)	9.13 (2.89)	9.22 (2.19)	10.76 (2.77)	**11.21** (0.69)
NN-RL (2-hidden)	0.44 (1.88)	1.38 (1.23)	2.36 (1.68)	3.48 (1.78)
Bilinear (2-hidden)	- - -	8.88 (1.75)	9.74 (1.86)	**11.28** (1.85)
Concatenation + DNN (2-hidden)	8.39 (2.07)	8.65 (1.35)	9.31 (0.44)	9.80 (2.45)

Table 3: The final average rewards and standard deviations on “Machine of Death”. NN-RL refers to using single state-action pair as input and single Q-value as output. “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 + DNN” refers to computing the Q function using a DNN with the concatenation of state and action embeddings as input.

of hidden layers. However, deep models also converge more slowly than 1 hidden layer DRRN or DQN models, as shown in Figure 6. Additionally, in Table 3, we added a baseline NN-RL, which takes a state-action pair and output a single Q-value. The results are worse than the modified DQN baselines, and are significantly worse than our proposed DRRN.

We also experimented with more complex interaction functions: a) a bilinear operation instead of an inner product with different action side dimensions; and b) a non-linear deep neural network using the concatenated state and action space embeddings as input and trained in an end-to-end fashion to predict Q values. The bilinear operation gave similar results, but the nonlinear function degraded performance.

In Appendix E, we show examples with pairs of state/actions texts, and their corresponding Q-values from a trained DRRN. Actions that are more likely to result in good endings are learned with high Q-values. For example, given “As you move forward, the people surrounding you suddenly look up with terror in their faces, and flee the street.”, a better action is to “Look up” rather than “Ignore the alarm of others and continue moving forward.” We also have anecdotal examples to show that the model has generalization ability, by using a well-trained DRRN to predict the Q-values for some made-up actions that were not included in the feasible set (Appendix F). This shows that our method has some generalization ability and can gain a reasonable level of language understanding in the game scenario.

5 Discussion and Future Work

In this paper we develop a deep reinforcement relevance network for the situation of an unbounded action space. Our contribution is the design of a novel model architecture for handling a variable number of actions defined through natural language in a text games setting. In addition, we show that the DRRN converges faster and to a better solution than the baseline deep Q-networks, using fewer parameters. Future work includes: (i) adding an attention model to robustly analyze which part of state/actions text correspond to strategic planning, and (ii) applying the proposed methods to more complex games or other tasks with unbounded action spaces.

References

Adams, Ernest. *Fundamentals of game design.* Pearson Education, 2014.

Bengio, Yoshua, Ducharme, Réjean, Vincent, Pascal, and Janvin, Christian. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

lower than highest assigned reward (30) because the game is stochastic (the optimal policy is not guaranteed to reach the best ending every time). 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.

Branavan, S.R.K., Chen, Harr, Zettlemoyer, Luke, and Barzilay, Regina. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 82–90, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P09/P09-1010`.

Branavan, SRK, Silver, David, and Barzilay, Regina. 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*, pp. 268–277. Association for Computational Linguistics, 2011.

Collobert, Ronan and Weston, Jason. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 160–167. ACM, 2008.

Dahl, George E, Yu, Dong, Deng, Li, and Acero, Alex. 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.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. 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.

Huang, Po-Sen, He, Xiaodong, Gao, Jianfeng, Deng, Li, Acero, Alex, and Heck, Larry. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pp. 2333–2338. ACM, 2013.

Kiros, Ryan, Zhu, Yukun, Salakhutdinov, Ruslan, Zemel, Richard S, Torralba, Antonio, Urtasun, Raquel, and Fidler, Sanja. Skip-thought vectors. *arXiv preprint arXiv:1506.06726*, 2015.

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

Le, Quoc V and Mikolov, Tomas. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *Nature*, 521(7553):436–444, 2015.

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

Lillicrap, Timothy P, Hunt, Jonathan J, Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, Silver, David, and Wierstra, Daan. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Lin, Long-Ji. Reinforcement learning for robots using neural networks. Technical report, DTIC Document, 1993.

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

Mnih, Andriy and Hinton, Geoffrey. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pp. 641–648. ACM, 2007.

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

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Narasimhan, Karthik, Kulkarni, Tejas, and Barzilay, Regina. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1–11, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL `http://aclweb.org/anthology/D15-1001`.

Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543, 2014.

Richard, Bellman. *Dynamic programming*. Princeton University Press, 1957.

Scheffler, Konrad and Young, Steve. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of the second international conference on Human Language Technology Research*, pp. 12–19. Morgan Kaufmann Publishers Inc., 2002.

Singh, Satinder P, Kearns, Michael J, Litman, Diane J, and Walker, Marilyn A. Reinforcement learning for spoken dialogue systems. In *Nips*, pp. 956–962, 1999.

Sutton, Richard S and Barto, Andrew G. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Tesauro, Gerald. Temporal difference learning and td-gammon. *Communications of the ACM*, 38 (3):58–68, 1995.

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

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

Table 4: Percentage of choice-based and hypertext-based text games since 2010, in archive of interactive fictions

A Percentage of Choice-based and Hypertext-based Text Games

As shown in Table 4.[9]

B Back Propagation Formula for Learning 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.

Forward:

$$h_{1,s} = f(W_{1,s}s_t + b_{1,s}) \tag{12}$$

$$h^i_{1,a} = f(W_{1,a}a^i_t + b_{1,a}), \quad i = 1, 2, 3, ..., |\mathcal{A}_t| \tag{13}$$

$$h_{l,s} = f(W_{l-1,s}h_{l-1,s} + b_{l-1,s}), \quad l = 2, 3, ..., L \tag{14}$$

$$h^i_{l,a} = f(W_{l-1,a}h^i_{l-1,a} + b_{l-1,a}), \quad i = 1, 2, 3, ..., |\mathcal{A}_t|, l = 2, 3, ..., L \tag{15}$$

$$Q(s_t, a^i_t) = h^T_{L,s}h^i_{L,a} \tag{16}$$

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.

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 $\Delta = [Q(s_t, a_t) - (r_t + \gamma \max_a Q(s_{t+1}, a))]^2/2$. Denote $\delta_{l,s} = \delta b_{l,s} = \partial Q/\partial b_s$, $\delta_{l,a} = \delta b_{l,a} = \partial Q/\partial b_a$, and we have (by following chain rules):

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

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

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

$$\begin{cases} \delta W_{1,s} = \partial Q/\partial W_{1,s} = \delta_{1,s} \cdot s^T_t \\ \delta W_{l,s} = \partial Q/\partial W_{l,s} = \delta_{l,s} \cdot h^T_{l-1,s}, \quad l = 2, 3, ..., L \end{cases}$$

$$\begin{cases} \delta W_{1,a} = \partial Q/\partial W_{1,a} = \delta_{1,a} \cdot a^T_t \\ \delta W_{l,a} = \partial Q/\partial W_{l,a} = \delta_{l,a} \cdot h^T_{l-1,a}, \quad l = 2, 3, ..., L \end{cases}$$

where $\odot$ denotes element-wise Hadamard product.

C Final Rewards in the Two Text Games

As shown in Table 5 and Table 6.

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

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 5: Final rewards defined for the text game "Saving John"

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 6: 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.

D Game 2 Learning curve with shared state and action embedding

As shown in Figure 7. For the first 1000 episodes, parameter tying gives faster convergence, but learning curve also has high variance and unstable.

E Examples of State-Action Pairs in the Two Text Games

As shown in Table 7 and Table 8.

F Examples of State-Action Pairs that do not exist in the feasible set

As shown in Table 9.

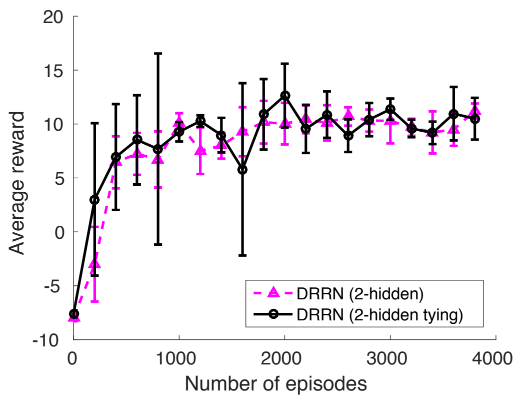


Figure 7: Learning curves of shared state-action embedding vs. proposed DRRN in Game 2

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 7: 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

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

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

Table 9: Q values (in parentheses) for state-action pair from "Machine of Death", using trained DRRN, with made-up actions that were not in the feasible set