Deep Reinforcement Learning with a Natural Language Action Space

Ji He*, **Jianshu Chen**†, **Xiaodong He**†, **Jianfeng Gao**†, **Lihong Li**†
Li Deng† **and 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.com

Abstract

This paper introduces a novel architecture for reinforcement learning with deep neural networks designed to handle state and action spaces characterized by natural language, as found in text-based games. Termed a deep reinforcement relevance network (DRRN), the architecture represents action and state spaces with separate embedding vectors, which are combined with an interaction function to approximate the Q-function in reinforcement learning. We evaluate the DRRN on two popular text games, showing superior performance over other deep Q-learning architectures. Experiments with paraphrased action descriptions show that the model is extracting meaning rather than simply memorizing strings of text.

1 Introduction

This work is concerned with learning strategies for sequential decision-making tasks, where a system takes actions at a particular state with the goal of maximizing a long-term reward. More specifically, we consider tasks where both the states and the actions are characterized by natural language, such as in human-computer dialog systems, tutoring systems, or text-based games. In a text-based game, for example, the player (or system, in this case) is given a text string that describes the current state of the game and several text strings that describe possible actions one could take. After selecting one of the actions, the environment state is updated and revealed in a new textual description. A reward is given either at each transition or in the end. The objective is to understand, at each step, the state text and all the action texts to pick the most relevant action, navigating through the sequence of texts so as to obtain the highest long-term reward. Here the notion of relevance is based on the joint state/action 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 a higher long-term reward. Because a player's action changes the environment, reinforcement learning (Sutton and Barto, 1998) is appropriate for modeling long-term dependency in text games.

There is a large body of work on reinforcement learning. Of most interest here are approaches leveraging neural networks because of their success in handling a large state space. Early work — TD-gammon — used a neural network to approximate the state value function (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. Building on the approach of Q-learning (Watkins and Dayan, 1992), the "Deep Q-Network" (DQN) was developed and applied to Atari games (Mnih et al., 2013; Mnih et al., 2015) and shown to achieve human level performance by applying convolutional neural networks to the raw image pixels. Narasimhan et al. (2015) applied a Long Short-Term Memory network to characterize the state space in a DQN framework for learning control policies for parser-based text games. More recently, Nogueira and Cho (2016) have also proposed a goal-driven web navigation task for language based sequential decision making study. Another stream of work focuses on continuous control with deep reinforcement learning (Lillicrap et al., 2016), where an actor-critic algorithm operates over a known continuous action space.

Inspired by these successes and recent work using neural networks to learn phrase- or sentence-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1621–1630,
Berlin, Germany, August 7-12, 2016.

level embeddings (Collobert and Weston, 2008; Huang et al., 2013; Le and Mikolov, 2014; Sutskever et al., 2014; Kiros et al., 2015), we propose a novel deep architecture for text understanding, which we call a deep reinforcement relevance network (DRRN). The DRRN uses separate deep neural networks to map state and action text strings into embedding vectors, from which "relevance" is measured numerically by a general interaction function, such as their inner product. The output of this interaction function defines the value of the Q-function for the current state-action pair, which characterizes the optimal long-term reward for pairing these two text strings. The Q-function approximation is learned in an end-to-end manner by Q-learning.

The DRRN differs from prior work in that earlier studies mostly considered action spaces that are bounded and known. For actions described by natural language text strings, the action space is inherently discrete and potentially unbounded due to the exponential complexity of language with respect to sentence length. A distinguishing aspect of the DRRN architecture — compared to simple DQN extensions — is that two different types of meaning representations are learned, reflecting the tendency for state texts to describe scenes and action texts to describe potential actions from the user. We show that the DRRN learns a continuous space representation of actions that successfully generalize to paraphrased descriptions of actions unseen in training.

2 Deep Reinforcement Relevance Network

2.1 Text Games and Q-learning

We consider the sequential decision making problem for text understanding. 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 attempts 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 the best action. Then, the environment state is updated $s_{t+1} = s'$ according to the probability $p(s'|s, a)$, and the agent receives 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. Define the Q-function $Q^\pi(s, a)$ as the expected return starting from s, taking the action a, and thereafter following policy $\pi(a|s)$ to be:

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

where γ denotes a discount factor. The optimal policy and Q-function can be found by using the Q-learning algorithm (Watkins and Dayan, 1992):

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \eta_t \cdot \left(r_t + \gamma \cdot \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right) \quad (1)$$

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:

$$\pi(a_t = a_t^i|s_t) = \frac{\exp(\alpha \cdot Q(s_t, a_t^i))}{\sum_{j=1}^{|\mathcal{A}_t|} \exp(\alpha \cdot Q(s_t, a_t^j))}, \quad (2)$$

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. α is kept constant throughout the learning period. All methods are initialized with small random weights, so initial Q-value differences will be small, thus making the Q-learning algorithm more explorative initially. 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 exploration probability.

2.2 Natural language 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 (1) 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 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 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 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 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.

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

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.

$$h_{1,s} = f(W_{1,s}s_t + b_{1,s}) \quad (3)$$

$$h_{1,a}^i = f(W_{1,a}a_t^i + b_{1,a}) \quad (4)$$

$$h_{l,s} = f(W_{l-1,s}h_{l-1,s} + b_{l-1,s}) \quad (5)$$

$$h_{l,a}^i = f(W_{l-1,a}h_{l-1,a}^i + b_{l-1,a}) \quad (6)$$

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:

$$Q(s, a^i; \Theta) = g\left(h_{L,s},\ h_{L,a}^i\right) \quad (7)$$

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. 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 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).

2.4 Learning the DRRN: Back propagation

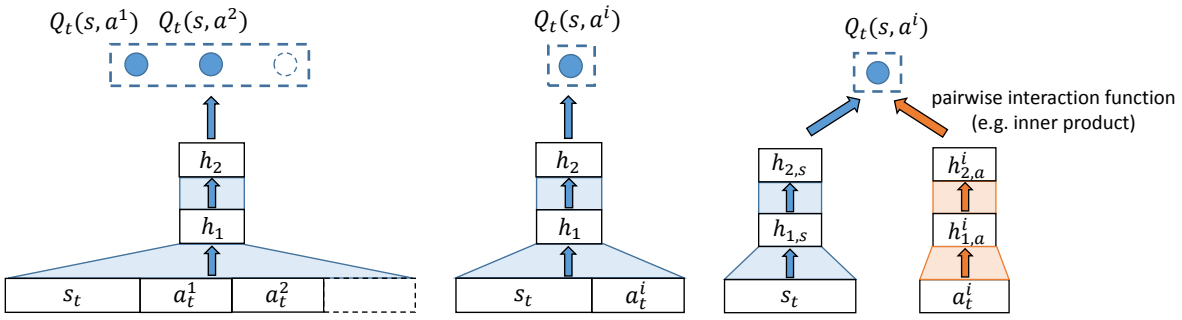
To learn the DRRN, we use the "experience-replay" strategy (Lin, 1993), which uses a fixed exploration policy to interact with the environment to obtain a sample trajectory. Then, we randomly sample a transition tuple (s_k, a_k, r_k, s_{k+1}), compute the temporal difference error for sample k:

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

and update the model according to the recursions:

$$W_{v,k} = W_{v,k-1} + \eta_k d_k \cdot \frac{\partial Q(s_k, a_k; \Theta_{k-1})}{\partial W_v} \quad (8)$$

$$b_{v,k} = b_{v,k-1} + \eta_k d_k \cdot \frac{\partial Q(s_k, a_k; \Theta_{k-1})}{\partial b_v} \quad (9)$$



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

Figure 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.

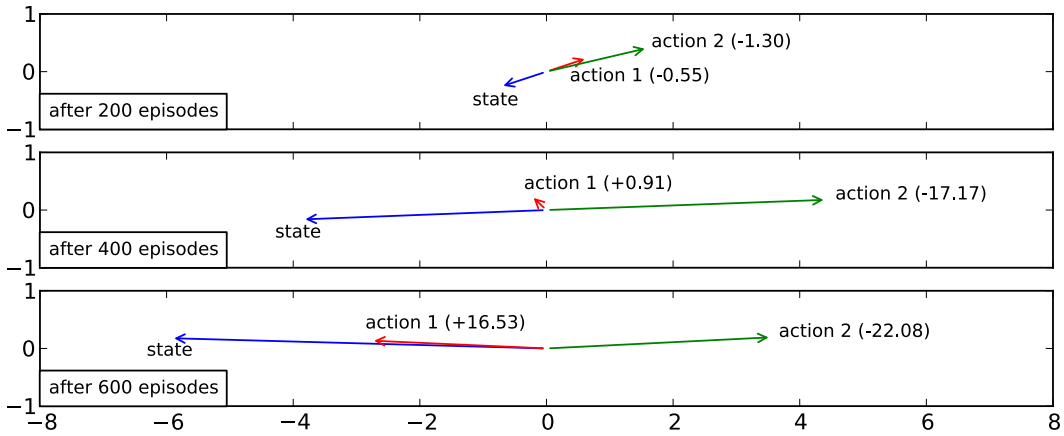


Figure 2: PCA projections of text embedding vectors for state and associated action vectors after 200, 400 and 600 training episodes. The state is "As you move forward, the people surrounding you suddenly look up with terror in their faces, and flee the street." Action 1 (good choice) is "Look up", and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

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 supplementary materials. 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 (Mnih et al., 2015). 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.

Figure 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.

3 Experimental Results

3.1 Text games

Text games, although simple compared to video games, still enjoy high popularity in online communities, with annual competitions held online

Algorithm 1 Learning algorithm for DRRN

Initialize replay memory $\mathcal{D}$ to capacity N.
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 $a_1^1, a_1^2, \ldots, a_1^{|\mathcal{A}_1|}$.
  for $t = 1, \ldots, T$ **do**
    Compute $Q(s_t, a_t^i; \Theta)$ for the list of actions using DRRN forward activation (Section 2.3).
    Select an action a_t based on probability distribution $\pi(a_t = a_t^i | s_t)$ (Equation 2)
    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 $(s_t, a_t, r_t, s_{t+1}, A_{t+1})$ in $\mathcal{D}$.
    Sample random mini batch of transitions $(s_k, a_k, r_k, s_{k+1}, A_{k+1})$ 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 $(y_k - Q(s_k, a_k; \Theta))^2$ with respect to the network parameters Θ (Section 2.4). Back-propagation is performed only for a_k even though there are $|\mathcal{A}_k|$ actions at time k.
  end for
end for

since 1995. Text games communicate to players in the form of a text display, which players have to understand and respond to by typing or clicking text (Adams, 2014). There are three types of text games: parser-based (Figure 3(a)), choice-based (Figure 3(b)), and hypertext-based (Figure 3(c)). Parser-based games accept typed-in commands from the player, usually in the form of verb phrases, such as "eat apple", "get key", or "go east". They involve the least complex action language. Choice-based and hypertext-based games present actions after or embedded within the state text. The player chooses an action, 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 more popular, increasing in percentage from 8% in 2010 to 62% in 2014.[1]

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. Thus the parser-based text game is reduced to a problem that is well suited to a fixed-action-set DQN. 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.

Game	Saving John	Machine of Death
Text game type	Choice	Choice & Hypertext
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

Table 1: Statistics for the games "Saving John" and and "Machine of Death".

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.[2] The 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". We manually annotate fi-

[1] Statistics obtained from `http://www.ifarchive.org`

[2] Simulators are available at `https://github.com/jvking/text-games`

(a) Parser-based (b) Choiced-based (c) Hypertext-based

Figure 3: Different types of text games

nal rewards for all distinct endings in both games (as shown in supplementary materials). 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, to encourage the learner to finish the game as soon as possible. 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. For text input, we use raw bag-of-words as features, with different vocabularies for the state side and action side.

3.2 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. We use DRRNs with 20, 50 and 100-dimension hidden layer(s) 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 text-based state and action descriptions, each 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, the highest scoring available action is used. The PA DQN baseline takes each pair of state-action texts as input, and generates a corresponding Q-value.

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)

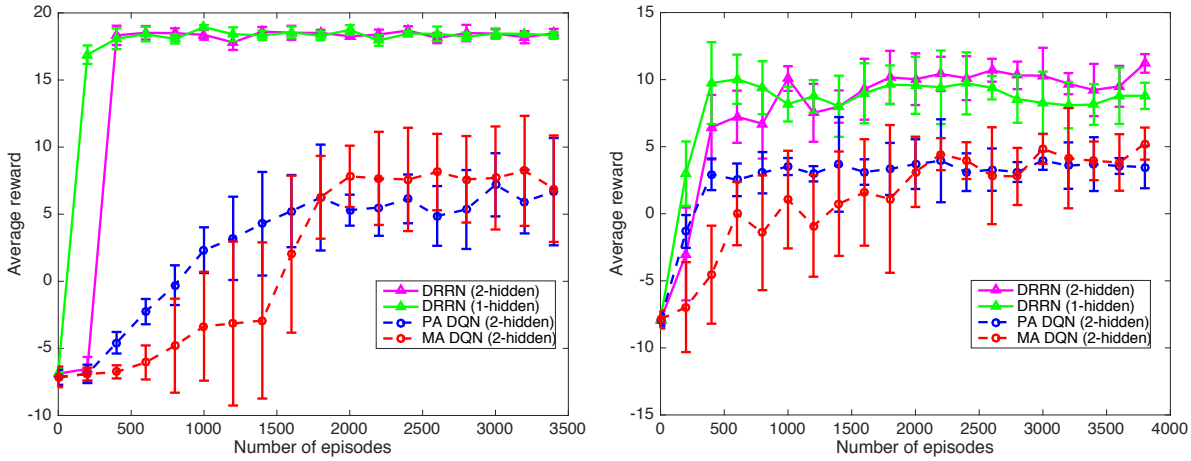
Table 2: The final average rewards and standard deviations on "Saving John".

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 (2), 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 2.4. 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.[3] This will encourage the algorithms to make decisions based on the understanding of the texts that describe the states and actions.

3.3 Performance

In Figure 4, we show the learning curves of different models, where the dimension of the hid-

[3]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.



(a) Game 1: "Saving John" (b) Game 2: "Machine of Death"

Figure 4: Learning curves of the two text games.

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

den 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 Game 1, and roughly -8 average rewards for Game 2), 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).

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 20, 50, and 100 dimensions in the hidden layers. The DRRN performs consistently better than all baselines, and often with a lower variance. For Game 2, 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 versions, as shown for the DRRN in Figure 4.

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 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. For different configurations, we fix the state side embedding to be 100 dimensions and vary the action side embedding dimensions. The bilinear operation gave similar results, but the concatenation input to a DNN degraded performance. Similar behaviors have been observed on a different task (Luong et al., 2015).

3.4 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 in 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%,

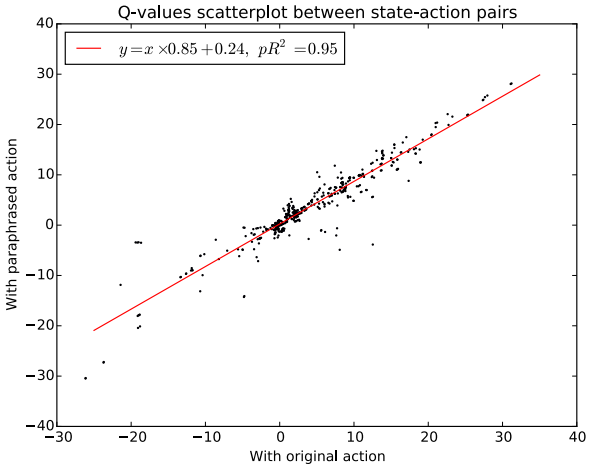


Figure 5: Scatterplot and strong correlation between Q-values of paraphrased actions versus original actions

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.

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 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, 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 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 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 proposed DRRN model trained with the original action descriptions for "Machine of Death", and test on paraphrased action descriptions. For this game, the underlying state transition mechanism has not changed. 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 5. 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 Related Work

There has been increasing interest in applying deep reinforcement learning to a variety problems, but only a few studies address problems with natural language state or action spaces. 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 and Young, 2002; Young et al., 2013). There has also been interest in extracting textual knowledge to improve game control performance (Branavan et al., 2011), and mapping text instructions to sequences of executable actions (Branavan et al., 2009). In some applications, it is possible to manually design features for state-action pairs, which are then used in reinforcement learning to learn a near-optimal policy (Li et al., 2009). Designing such features, however, require substantial domain knowledge.

The work most closely related to our study involves application of deep reinforcement to learning decision policies for parser-based text games. Narasimhan et al. (2015) applied a Long Short-Term Memory DQN framework, which achieves higher average reward than the random and Bag-of-Words DQN baselines. In this work, actions are constrained to a set of known fixed command structures (one action and one argument object),

	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: 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 5: The final average rewards and standard deviations on paraphrased game "Machine of Death".

based on a limited action-side vocabulary size. The overall action space is defined by the action-argument product space. This pre-specified product space is not feasible for the more complex text strings in other forms of text-based games. Our proposed DRRN, on the other hand, can handle the more complex text strings, as well as parser-based games. In preliminary experiments with the parser-based game from (Narasimhan et al., 2015), we find that the DRRN using a bag-of-words (BOW) input achieves results on par with their BOW DQN. The main advantage of the DRRN is that it can also handle actions described with more complex language.

The DRRN experiments described here leverage only a simple bag-of-words representation of phrases and sentences. As observed in (Narasimhan et al., 2015), more complex sentence-based models can give further improvements. In preliminary experiments with "Machine of Death", we did not find LSTMs to give improved performance, but we conjecture that they would be useful in larger-scale tasks, or when the word embeddings are initialized by training on large data sets.

As mentioned earlier, other work has applied deep reinforcement learning to a problem with a continuous action space (Lillicrap et al., 2016). In the DRRN, the action space is inherently discrete, but we learn a continuous representation of it. As indicated by the paraphrasing experiment, the continuous space representation seems to generalize reasonably well.

5 Conclusion

In this paper we develop a deep reinforcement relevance network, a novel DNN architecture for handling actions described by natural language in decision-making tasks such as text games. 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. 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 text games or other tasks with actions defined through natural language.

Acknowledgments

We thank Karthik Narasimhan and Tejas Kulkarni for providing instructions on setting up their parser-based games.

References

E. Adams. 2014. *Fundamentals of game design*. Pearson Education.

S.R.K. Branavan, H. Chen, L. Zettlemoyer, and R. Barzilay. 2009. 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.

S.R.K. Branavan, D. Silver, and R. Barzilay. 2011. Learning to win by reading manuals in a monte-carlo framework. In *Proc. of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 268–277. Association for Computational Linguistics.

R. Collobert and J. Weston. 2008. 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.

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

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. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.*, 29(6):82–97.

P-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. 2013. 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.

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

A. Krizhevsky, I. Sutskever, and G. E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Q. V Le and T. Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*.

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

L. Li, J. D. Williams, and S. Balakrishnan. 2009. 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.

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

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

M-T. Luong, H. Pham, and C. D. Manning. 2015. 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.

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

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

K. Narasimhan, T. Kulkarni, and R. Barzilay. 2015. 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.

R. Nogueira and K. Cho. 2016. Webnav: A new large-scale task for natural language based sequential decision making. *arXiv preprint arXiv:1602.02261*.

K. Scheffler and S. Young. 2002. 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.

I. Sutskever, O. Vinyals, and Q. V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

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

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

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

S. Young, M. Gasic, B. Thomson, and J. D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.