Thread Popularity Prediction and Tracking with a Permutation-invariant Model

Hou Pong Chan[1,2] **and Irwin King**[1,2]
[1]Department of Computer Science and Engineering,
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[2]Shenzhen Key Laboratory of Rich Media Big Data Analytics and Application,
Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China
{hpchan, king}@cse.cuhk.edu.hk

Abstract

The task of thread popularity prediction and tracking aims to recommend a few popular comments to subscribed users when a batch of new comments arrive in a discussion thread. This task has been formulated as a reinforcement learning problem, in which the reward of the agent is the sum of positive responses received by the recommended comments. In this work, we propose a novel approach to tackle this problem. First, we propose a deep neural network architecture to model the expected cumulative reward (Q-value) of a recommendation (action). Unlike the state-of-the-art approach, which treats an action as a sequence, our model uses an attention mechanism to integrate information from a set of comments. Thus, the prediction of Q-value is invariant to the permutation of the comments, which leads to a more consistent agent behavior. Second, we employ a greedy procedure to approximate the action that maximizes the predicted Q-value from a combinatorial action space. Different from the state-of-the-art approach, this procedure does not require an additional pre-trained model to generate candidate actions. Experiments on five real-world datasets show that our approach outperforms the state-of-the-art.

1 Introduction

Online discussion forums allow people to join in-depth conversations about different topics in form of threads. Each thread corresponds to one conversation, which is initiated by a post and users respond to it with comments. In addition, a comment can be further replied by another comment, forming a discussion tree. Users who are interested in a particular thread will subscribe to it. After the subscription, users will receive a notification when a new comment arrives in that thread. However, the speed of content generation in a well-known discussion forum is breakneck. For instance, in Reddit[1], there were more than 900 million comments posted in 2017 (Reddit, 2017). Hence, merely pushing every new comment to the subscribers leads to a poor user experience. Motivated by this issue, He et al. (2016c) proposed the task of thread *popularity prediction* and *tracking*. When N new comments arrive in a thread, the system performs one step of recommendation by pushing K comments to the subscribers. We want to maximize the sum of popularities of the recommended comments over all recommendation steps. The popularity of a comment is measured by the number of positive reactions it received, e.g., the rating. With the assumption that a user needs to know the prior context in order to understand a comment, the system can only recommend new comments that are in the subtrees of previously recommended comments. Thus, the selection of comments at the current recommendation step will affect the comments that we can choose in the future recommendation steps.

To incorporate the long-term consequences of recommendations, the task of thread popularity prediction and tracking has been formulated as a reinforcement learning problem, in which an agent selects an action (a set of K comments) according to its current state (previous recommended comments), with the goal of maximizing the cumulative reward (total popularities of the recommended comments over all recommendation steps). The optimal action of the agent at each step is the action that maximizes the Q-function, $Q(s, a)$, which denotes the long-term reward of choosing action a in state s. In practice, we learn this Q-function using a parametric function, $Q(s, a; \theta)$, where θ is the model parameter vector. Thus, the predicted optimal action of the agent is the action

[1]https://www.reddit.com/

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3392–3401
Brussels, Belgium, October 31 - November 4, 2018.

that maximizes $Q(s, a; \theta)$.

This reinforcement learning problem has two main challenges. First, we need to develop a parametric model, $Q(s, a; \theta)$, to approximate the Q-function. Second, finding the action that maximizes $Q(s, a; \theta)$ requires the prediction of all $\binom{N}{K}$ possible actions, which is intractable. Thus, we need a procedure to approximate the predicted optimal action from a combinatorial action space.

To address the first challenge, He et al. (2016c) proposed a neural network model, DRRN-BiLSTM, to approximate the Q-function. In this model, a bi-directional long short-term memory (LSTM) (Graves and Schmidhuber, 2005) is used to encode the set of K comments in an action. To address the second challenge, they proposed the two-stage Q-learning procedure to approximate the predicted optimal action (He et al., 2017). In this procedure, the agent uses a pre-trained and less-sophisticated model to rank all possible actions, then it uses the DRRN-BiLSTM to re-rank the top-M actions and selects the best one. However, this approach has two limitations. First of all, bi-directional LSTM is a sequence model, which treats the set of K comments in an action as a sequence. Although they tried to fix this problem by feeding randomly-permuted comments to the model, a different permutation of the same set of comments leads to a different Q-value prediction. Thus, the agent may not consistently select the predicted optimal action. Secondly, the two-stage Q-learning procedure requires an additional pre-trained model to generate candidate actions.

Our work addresses these two limitations as follows. We propose a novel neural network model, DRRN-Attention, to approximate the Q-function. In our model, we use an attention mechanism (Bahdanau et al., 2014) to integrate the information from a set of comment into an action embedding vector. In a nutshell, the attention mechanism outputs a weighted sum of the comment representations, where the weights are learned by a subnetwork to indicate the importance of each comment. Thus, the action embedding is invariant to the permutation of the comments, which leads to a permutation invariant Q-value prediction. Next, we employ a greedy procedure to approximate the action that maximizes $Q(s, a; \theta)$. This procedure only requires the prediction of $O(NK)$ actions, which is significantly lower than $\binom{N}{K}$. Moreover, it does not require an additional pre-trained model to generate candidate actions.

In our experiments, we evaluate the performance of our DRRN-Attention model and the greedy approximation procedure against the baselines on five real-world datasets. Experimental results demonstrate that our approach beats the baselines on four of the datasets and achieves a competitive performance on one of the datasets. Furthermore, we analyze the performance of our approach across four action sizes ($K = 2, 3, 4, 5$). Our approach consistently achieves a higher cumulative reward than the baselines across all these action sizes.

We summarize our contributions as follow: (1) a new neural network architecture to model the Q-value of the agent which is invariant to the permutation of sub-actions; (2) a greedy procedure for the agent to select an action from the combinatorial action space without an additional pre-trained model; and (3) the new state-of-the-art performances on five real-world datasets.

2 Related Work

2.1 Reinforcement Learning in Text-based Tasks

Reinforcement learning has been widely applied in various text-based tasks. There are several articles in literature studying the tasks of mapping instruction manuals to a sequence of commands, such as game commands (Branavan et al., 2011), software commands (Branavan et al., 2010), and navigation directions (Vogel and Jurafsky, 2010). In the task of text-based game, an agent selects a textual command from a set of feasible commands at every time step. Narasimhan et al. (2016) considered a special case that all the textual commands have a fixed structure, while He et al. (2016b) and Chen et al. (2017) considered another case that all commands are free text.

In the task of thread popularity prediction and tracking, the agent selects a set of K comments from N available comments at every time step, where each comment is a free text. He et al. (2016c) proposed two different approaches to tackle this task. In their first approach, the agent uses the Deep Reinforcement Relevance Network (DRRN) (He et al., 2016b) to model the Q-function of selecting a comment. In their second approach, the agent uses the DRRN-BiLSTM (He et al., 2016c) to model the Q-function of an ac-

tion. To due with the combinatorial action space, the agent uses uniform sampling to generate a set of M candidate actions. To improve this random sampling scheme, they proposed the two-stage Q-learning procedure in their later work (He et al., 2017), which used a pre-trained model to generate M candidate actions. Their experimental results showed that using DRRN-BiLSTM with two-stage Q-learning procedure outperforms all other existing methods. The difference between our model and DRRN-BiLSTM is that we use attention to encode a set of comments rather than using a bi-directional LSTM. Besides, the greedy procedure in our approach does not require any extra pre-trained model. He et al. (2017) also considered a special case that the agent can access an external knowledge source to augment the state representation. This setting is orthogonal to this work since we focus on the action encoding and the approximation of predicted optimal action.

One line of research focused on the integration of sequence-to-sequence (SEQ2SEQ) model (Sutskever et al., 2014) and reinforcement learning framework, examples including dialogue generation (Dhingra et al., 2017; Li et al., 2016; Su et al., 2016), question answering system (Buck et al., 2017), and machine translation (He et al., 2016a). In these tasks, the agent selects an action by generating a free text using a SEQ2SEQ model.

2.2 Deep Learning on Sets

Most of the deep learning models on sets employed attention to integrate information from a set of input. This idea was first introduced in the read-process-and-write network (Vinyals et al., 2016), which uses a process module to perform multiple steps of attention over a set of vectors to obtain a permutation-invariant embedding. Our work adapts this idea to aggregate a set of comment embedding vectors. In the domain of graph learning, several models (Sukhbaatar et al., 2015; Zhang et al., 2017) learn an embedding of a node by attending over its neighboring nodes. All of the above models can be interpreted as a special case of memory network (Weston et al., 2015; Sukhbaatar et al., 2015; Zhang et al., 2017), if we view the set of feature vectors as external memory. Max-pooling is another promising technique for the problem of learning on sets. Qi et al. (2017) used max-pooling to aggregate the feature vectors of a set of 3D geometry points. Recently, Zaheer et al. (2017) derived the necessary and sufficient conditions for a neural network layer to be permutation invariant.

2.3 Popularity Prediction

Another related line of research is popularity prediction problem in a supervised learning setting. Yano and Smith (2010) used the LDA topic model (Blei et al., 2003) to predict the number of comments of a blog post in a political blog. There are also several studies focused on the task of predicting the number of reshares on Facebook (Cheng et al., 2014) and the number of retweets in tweeter based on the text content (Tan et al., 2014; Hong et al., 2011). Recently, Cheng et al. (2017) proposed a neural network model to learn comment embeddings for the task of community endorsement prediction in a supervised learning setting.

3 Preliminary

3.1 Discussion Tree

A discussion thread in an online forum can be represented as a tree. Each node in the tree stores a free text. The root node represents the post of the thread and each non-root node represents a comment of the thread. There is a directed edge from node u to node v if and only if a comment (or post) u is replied by comment v. This tree keeps growing as new comments are submitted to the thread.

3.2 Problem Definition

The task of thread popularity prediction and tracking is formally defined as a reinforcement learning problem. We use M_t to denote the set of comments that are being tracked at time t. Given a discussion thread, we start an episode as follows. First, we initialize M_1 to be the post of a thread. Then, at each time step t the agent performs the following operations:

- Read the current state s_t, which is all the previously tracked comments $\{M_1, \ldots, M_t\}$.
- Read N new comments, $c_t = \{c_{t,1}, \ldots, c_{t,N}\}$, in the subtree of M_t.
- Select a set of K comments from c_t to recommend, $a_t = \{c_t^1, \ldots, c_t^K\}$, where $c_t^i \in c_t$ for $i = 1, \ldots, K$ and $c_t^i \neq c_t^j$ for $i \neq j$.

- Receive a reward, $r_{t+1} = \sum_{i=1}^{K} \eta_{c_t^i}$, where $\eta_{c_t^i}$ is the number of positive reactions received by comment c_t^i.

- Track the set of recommended comments in the next time step, $M_{t+1} = a_t$.

The episode terminates when no more new comments appear in the subtree of M_t. The goal of the agent is to maximize the cumulative reward.

3.3 Q-function

The state-action value function (or Q-function), $Q(s, a)$, is defined as the expected cumulative reward starting from state s and taking action a. More formally, $Q(s, a) = \mathbb{E}[\sum_{l=0}^{+\infty} \gamma^l r_{t+1+l} | s_t = s, a_t = a]$, where $\gamma \in (0, 1]$ is a discount factor for future rewards. Since the goal of the agent is to maximize the cumulative reward, the optimal action for each state is the action that achieves the highest Q-value. Thus, the Q-function is associated with an optimal policy: in every state, the agent selects the action that maximizes the Q-function, i.e., $a_t = \text{argmax}_a Q(s_t, a)$, $\forall t$. Since this Q-function is unknown to the agent, we approximate the Q-function using a parametric model, $Q(s, a; \theta)$, and update the parameters θ using received rewards.

3.4 Exploration-Exploitation Trade-off

The agent needs to balance the exploration-exploitation trade-off when selecting an action. On one hand, the agent can choose the action with the highest estimated Q-value to exploit its current knowledge of the Q-function. On the other hand, the agent can choose a non-greedy action to get more information about the Q-value of other actions. The balance between exploration and exploitation can be achieved by using the ϵ-greedy policy, in which the agent selects a random action with probability ϵ, and selects a greedy action with probability $1 - \epsilon$. Note that the term "greedy" in the ϵ-greedy policy means that the agent selects the action that is predicted to be optimal, i.e., select $a_t = \text{argmax}_a Q(s_t, a; \theta)$. It does not refer to the greedy procedure, which is used to approximate the predicted optimal action in a combinatorial action space.

4 DRRN-Attention Model

In this work, we propose a new deep neural network model, named DRRN-Attention, to approximate the Q-function for the task of thread popularity prediction and tracking. The input to our model is a state, s_t, and an action, $a_t = \{c_t^1, \ldots, c_t^K\}$, as defined in Section 3.2. The output is the prediction of Q-value, i.e., $Q(s_t, a_t; \theta) \in \mathbb{R}$. Figure 1 illustrates the overall architecture of DRRN-Attention. We divide our model into three modules as follows.

4.1 Text Representation Module

The text representation module reads s_t and $a_t = \{c_t^1, \ldots, c_t^K\}$. We first convert $s_t, c_t^1, \ldots, c_t^K$ into bag-of-words (BOW) representations, $b_{s_t}, b_{c_t^1}, \ldots, b_{c_t^K}$. Then, we use a 2-layer feedforward neural network to embed b_{s_t} into a d-dimensional state embedding vector, $m_{s_t} \in \mathbb{R}^d$. After that, we use another 2-layer feedforward neural network to embed $b_{c_t^i}$ into a d-dimensional comment embedding vector, $m_{c_t^i} \in \mathbb{R}^d$, for $i = 1, \ldots, K$. This module outputs m_{s_t} and $\{m_{c_t^1}, \ldots, m_{c_t^K}\}$.

4.2 Set Embedding Module

The input to this module is a set of d-dimensional comment embeddings, $\{m_{c_t^1}, \ldots, m_{c_t^K}\}$. The output is an action embedding vector, $m_{a_t} \in \mathbb{R}^{h+d}$, which is invariant to the ordering of comment embeddings. The module consists of a single-layer LSTM with a hidden size of h, and a shared attention mechanism, $f : \mathbb{R}^h \times \mathbb{R}^d \rightarrow \mathbb{R}$. The initial hidden state, $q_0 \in \mathbb{R}^h$, of the LSTM is a trainable vector. Inspired by (Vinyals et al., 2016), we perform L steps of computations over the comment embedding vectors. More specifically, at each step of computation $l = 0, 1, \ldots, L-1$:

- The query vector, $q_l \in \mathbb{R}^h$, is the current hidden state of the LSTM.

- Apply the attention mechanism to compute an attention coefficient, $e_{i,l}$, between the query, q_l, and a comment embedding, $m_{c_t^i}$, for $i = 1, \ldots, K$. In general, this framework is agnostic to the underlying attention mechanism. In this work, we closely follow the attentional setup in (Bahdanau et al., 2014), as shown in the following equation.

$$e_{i,l} = v^T \tanh(\mathbf{W_e} m_{c_t^i} + \mathbf{U_e} q_l), \qquad (1)$$

where $v \in \mathbb{R}^{h'}$, $\mathbf{W}_e \in \mathbb{R}^{h' \times d}$, and $\mathbf{U}_e \in \mathbb{R}^{h' \times h}$.

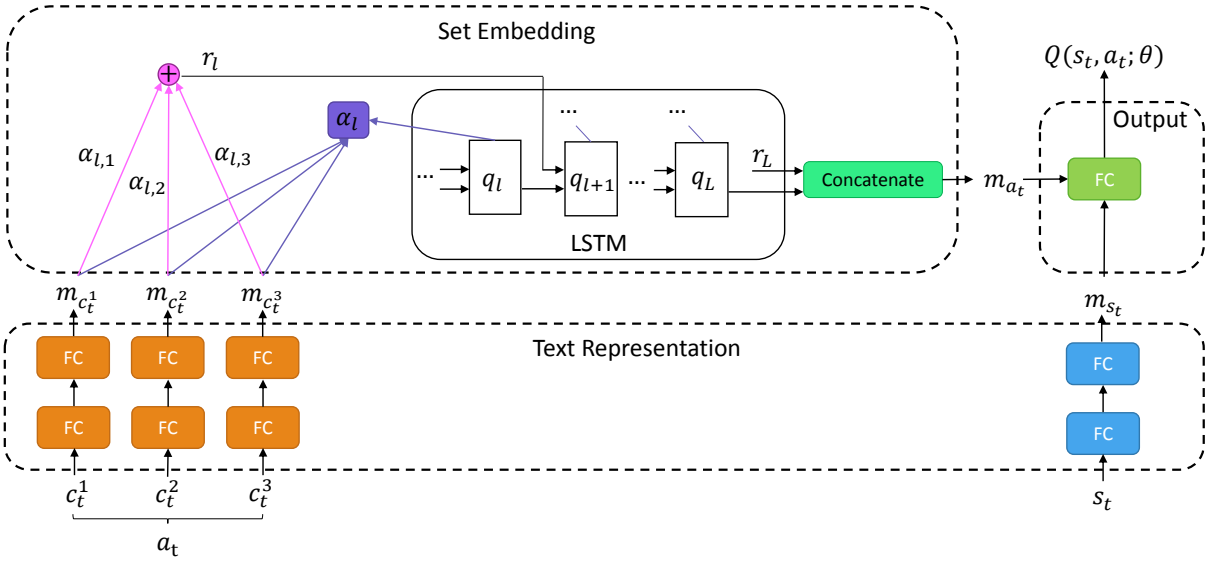

Figure 1: Architecture of DRRN-Attention model. The text representation module first converts state s_t and each comment c_t^i in action a_t into embedding vectors. Then, the comment embedding vectors are passed to the set embedding module to learn an action embedding vector. Finally, the state embedding and the action embedding are passed to the output module to output a prediction of Q-value.

- Apply softmax function to normalize the attention coefficients,

$$\alpha_{i,l} = \frac{\exp(e_{i,l})}{\sum_{j=1}^{K} \exp(e_{j,l})}. \quad (2)$$

- Use the normalized attention coefficients to compute a weighted sum of the comment embedding vectors, as the readout in this computation step,

$$r_l = \sum_{i=1}^{K} \alpha_{i,l} m_{c_t^i}. \quad (3)$$

- The LSTM takes q_l and r_l as input and computes the next hidden state, q_{l+1},

$$q_{l+1} = \text{LSTM}([q_l, r_l]). \quad (4)$$

Note that swapping any two comment embedding vectors, $m_{c_t^i}$ and $m_{c_t^j}$, will not affect the query vector q_l as well as the attention readout r_l. After L steps of computation, this module concatenates q_L and r_L to yield the final output action embedding, $m_{a_t} = [q_L, r_L] \in \mathbb{R}^{h+d}$.

4.3 Output Module

The input to this module is a state embedding vector, $m_{s_t} \in \mathbb{R}^d$, and an action embedding vector, $m_{a_t} \in \mathbb{R}^{h+d}$. We simply concatenate m_{s_t} and m_{a_t} and pass them through a fully-connected layer. The output is $Q(s_t, a_t; \theta) \in \mathbb{R}$, which is the prediction of $Q(s_t, a_t)$.

Algorithm 1 Greedy$(s_t, c_t, Q(\cdot, \cdot; \theta), K)$

```
a = ∅
for i = 1 → K do
    c* = argmax_{c ∈ c_t \ a} Q(s_t, a ∪ c; θ)
    a = a ∪ c*
return a
```

5 Greedy Procedure

The next challenge that we need to address is to approximate the predicted optimal action in a combinatorial action space. Finding the predicted optimal action, $\text{argmax}_a Q(s_t, a; \theta)$, is intractable since it requires the prediction of all $\binom{N}{K}$ actions. In this work, we use a greedy procedure to compute an approximation. The complete procedure is shown in Algorithm 1. We start from an empty action, $a_t = \emptyset$, and then iteratively adds into a_t the comments that leads to the largest increase in $Q(s_t, a_t; \theta)$, until $|a_t| = K$. The procedure consists of K iterations. In each iteration i, we need to predict the Q-value for $n - i$ actions. In total, it only requires the prediction of $O(NK)$ actions, which is tractable. The advantage of this procedure over existing methods is that it does not require another pre-trained model to generate candidate actions.

6 Parameter Learning

We use the deep Q-learning algorithm (Mnih et al., 2015), which is a variant of the traditional Q-

learning algorithm (Watkins and Dayan, 1992), to learn the model parameters of $Q(s, a; \theta)$ from the received rewards. The complete training procedure is shown in Algorithm 2 in the Appendix. The network parameters θ are first initialized arbitrarily. At each time step t, the agent selects an action a_t according to the ϵ-greedy policy, receives a reward r_{t+1}, and transits to the next state s_{t+1}. Thus, it yields a transition tuple, $\zeta_t = (s_t, a_t, r_{t+1}, s_{t+1})$. Instead of using the current transition tuple, ζ_t, to update the parameters, we first store ζ_t into an experience memory, D. This experience memory has a limited capacity, $|D|$, and the stored transition tuples are rewritten in a first-in-first-out manner. Then, we sample mini-batches of transition tuples (s, a, r, s') from D uniformly at random. Using the sampled transition tuples, we perform a step of stochastic gradient descent to minimize the following loss function,

$$L(\theta) = \mathbb{E}_{(s,a,r,s') \sim U(D)}[(y - Q(s, a; \theta))^2], \quad (5)$$

where $y = r + \gamma \max_{a'} Q(s', a'; \theta^-)$ is the Q-learning target, θ^- are the network parameters of the Q-learning target. We update θ^- to match the network parameters, θ, after every F time steps, where F is a hyperparameter.

7 Experimental Setup

In the experiments, we analyze the performances of different neural network models and different approximation procedures. First, their performances are evaluated on five real-world datasets, with a fix action size K. Then, we evaluate their performances with different action sizes, on one dataset. For each experiment setting, we do the following comparative analysis:

- Compare the performance of our DRRN-Attention model with the baseline models using different approximation procedures.
- Compare the performance of the greedy procedure with the baseline approximation procedures using different neural network models.
- Find the approach (combination of neural network model and approximation procedure) that achieves the best performance.

Finally, we conduct a case study to better illustrate the difference between our DRRN-Attention model and the DRRN-BiLSTM baseline.

Subreddit	# Posts	# Comments
askscience	0.94k	0.15M
askmen	4.45k	0.94M
todayilearned	9.44k	4.65M
worldnews	8.00k	4.28M
nfl	11.73k	5.72M

Table 1: Basic statics of discussion threads data from five subreddits.

7.1 Datasets

All the experiments are conducted on discussion thread data from the Reddit discussion forum. In Reddit, threads are grouped into different categories, called subreddits, according to different discussion themes. Registered users are allowed to give up-votes or down-votes to a comment, these votes are then aggregated to compute a *karma score* for the comment. We use it as the reward for recommending that comment. Using the post IDs provided by He et al. (2016c), we crawl five datasets from five different subreddits respectively, including askscience, askmen, todayilearned, worldnews, and nfl. These subreddits cover a wide range of discussion topics and language styles. The basic statistics of the datasets are presented in Table 1. Since some of the posts and comments were deleted by Reddit, we remove all the deleted posts and comments from the datasets. Thus, the statistics of our datasets are different from that in He et al. (2016c). For each dataset, we use the simulator provided by He et al. (2016c) to partition 90% of the data as a training set, and 10% of the data as a testing set.

7.2 Evaluation

The evaluation metric is the cumulative reward per episode averaged over 1,000 episodes (He et al., 2016c). For each setting, we evaluate an agent as follows. First, we train the agent on the training set using Algorithm 2 in the Appendix for 3,500 episodes. Then, we test the agent using the testing set for 1,000 episodes and choose every action according to the ϵ-greedy policy, but the agent cannot use the received rewards to update the model parameters. We repeat the testing for five repetitions and report the mean and the standard deviation of the evaluation metric. Throughout the training and testing, we fix $\epsilon = 0.1$.

Models	Procedure	Askscience	Askmen	Todayilearned	Worldnews	NFL
DRRN-BiLSTM	Random	546.9±21.8	183.1±7.2	654.9±27.3	427.4±21.9	438.7±16.1
	Two-stage	643.2±28.2	184.3±9.2	659.0±25.6	424.0±14.0	455.9±18.7
	Greedy	672.2±22.2	190.2±11.5	665.7±11.4	428.8±21.0	459.7±9.3
DRRN-Mean	Random	594.5±21.9	184.7±6.9	649.6±21.7	413.9±11.7	434.1±22.9
	Two-stage	581.1±13.0	183.3±5.6	670.4±29.7	426.7±24.2	441.9±20.3
	Greedy	732.3±25.7	194.2±7.8	680.0±31.0	435.4±18.5	452.3±22.6
Our model	Random	648.6±19.6	186.1±8.2	670.5±26.5	429.4±20.8	452.5±17.9
	Two-stage	685.6±24.6	184.7±7.5	672.2±37.8	426.1±18.2	**460.3±15.7**
	Greedy	**757.4±10.9**	**210.6±11.3**	**689.5±13.3**	**436.0±21.9**	454.2±12.0

Table 2: Comparison of average episodic reward on different datasets.

7.3 Baselines

Our DRRN-Attention model is compared with two baselines. The first one is DRRN-BiLSTM, which is the current state-of-the-art model to approximate the Q-value for this task (He et al., 2016c). We modify the DRRN-BiLSTM model by replacing the Bi-directional LSTM with a mean operator and call this new model DRRN-Mean. This DRRN-mean is used as the second baseline model. In addition, we compare the greedy procedure with two baseline approximation procedures. The first is random sampling procedure in (He et al., 2016c). The second is the two-stage Q-learning procedure in (He et al., 2017), which is the state-of-the-art approximation procedure for this task.

7.4 Implementation Details

In preprocessing, we remove all punctuations and lowercase all alphabetic characters. To construct the bag-of-words representations, we use the dictionary provided by He et al. (2016c). This dictionary contains the most frequent 5,000 words in the data. All model parameters are initialized by a uniform distribution within the interval $[-0.1, 0.1]$. In our DRRN-Attention model, we set the comment embedding size d to 16, the hidden size of the LSTM h to 16, the hidden size of the attention mechanism h' to 16, and the steps of attention L to 2. In the text embedding module of DRRN-Attention, each fully-connected layer has a hidden size of 16. In the baseline models, we set the hidden size of the bidirectional LSTM to 20, the comment embedding size to 20. The text embedding module of the baselines has two layers and each layer has a hidden size of 20. For the deep Q-learning algorithm, we set $F = 1000$. We update the model parameters using stochastic gradient descent with RMSprop (Tieleman and Hinton, 2012). We set different initial learning rates for different datasets (askscience: 0.00001; askmen: 0.00008; todayilearned, worldnews, and nfl: 0.00002). The mini batch size is 100. All the above hyperparameters are tuned by five-fold cross-validation. We also found that the model performances tuned by five-fold cross-validation are similar to that tuned by the testing set. We set the remaining hyperparameters according to (He et al., 2017). The memory size $|D|$ is set to 10000. The discount factor γ is set to 0.9. The candidate size m of the baseline approximation procedures is set to 10.

8 Experimental Results

8.1 Agent Performances on Various Datasets

In this section, the performances of different neural network models and approximation procedures are evaluated on five datasets with $N = 10$, $K = 3$. The results are shown in Table 2. We analyze the performances of our DRRN-Attention model in each approximation procedure. With the random sampling procedure, our model achieves a higher cumulative reward than the baseline models across all datasets. When using the two-stage Q-learning procedure, or the greedy procedure, our model outperforms the baselines on four of the datasets; its performance is competitive to the baselines in the remaining dataset. Next, we analyze the performances of the greedy procedure in each neural network model. When using the DRRN-BiLSTM model or the DRRN-Mean model to parameterize the Q-function, the greedy procedure achieves a higher cumulative reward than other two baseline procedures across all datasets. When using the DRRN-Attention model, the greedy procedure outperforms the baseline procedures in four of the datasets. In sum up, us-

Models	Procedure	K=2	K=3	K=4	K=5
DRRN-BiLSTM	Random	431.6±18.8	546.9±21.8	743.7±15.9	899.5±53.9
	Two-stage	484.3±10.6	643.2±28.2	765.7±32.8	928.6±20.5
	Greedy	467.8±24.5	672.2±22.2	772.9±28.2	929.2±18.6
Our model	Random	482.7±18.1	648.6±19.6	806.0±13.5	941.2±7.3
	Two-stage	537.5±26.1	685.6±24.6	772.8±22.8	903.5±22.4
	Greedy	**545.2±27.3**	**757.4±10.9**	**820.1±24.2**	**944.1±29.6**

Table 3: Comparison of average episodic reward with different action sizes on askscience dataset.

ing DRRN-Attention model with the greedy procedure outperforms all the baselines in four of the datasets.

8.2 Agent Performances on Various Action Sizes

We evaluate all the neural network models and approximation procedures across various action sizes with $K = 2, 3, 4, 5$ and fix $N = 10$ on the askscience dataset. The results are presented in Table 3. Our DRRN-Attention model outperforms the baselines across all the action sizes form $K = 2$ to $K = 5$ with the random sampling procedure or the greedy procedure. With the two-stage Q-learning procedure, our model achieves a higher cumulative reward than the baseline models when $K = 2, 3, 4$. Then, we analyze the performance of the greedy procedure in each neural network model. When using the DRRN-BiLSTM to parameterize the Q-function, the greedy procedure outperforms the baseline procedures when $K = 3, 4, 5$. When using our DRRN-Attention, the greedy procedure achieves a higher cumulative reward than the baselines when $K = 2, 3, 5$. Overall, using the DRRN-Attention model with the greedy procedure achieves the best performances across all the action sizes form $K = 2$ to $K = 5$.

8.3 Case Study

Table 4 presents an example of Q-value prediction of a state and three sub-actions on the askscience dataset. In this study, we enumerate every permutation of these three sub-actions, e.g., (1, 3, 2) denotes a permutation of comments that we place comment (1) in the first position, comment (3) in the second position, and comment (2) in the third position. Then, we use a trained DRRN-BiLSTM model and a trained DRRN-Attention model to predict the Q-value of each permutation of comments. When we use the DRRN-BiLSTM model, a different permutation of comments yields a different Q-value prediction. The Q-value prediction of the permutation (1, 3, 2) almost triples that of the permutation (2, 3, 1). On the other hand, any permutation of the comments does not change the Q-value prediction when we use our DRRN-Attention model. This example demonstrates that the ordering of the comments can significantly affect the predicted Q-value when we use the DRRN-BiLSTM model.

State		
Is the heat I feel when I face a bonfire transmitted to me mostly by infrared radiation or by heated air?		
Sub-actions (comments)		
(1) Should it also be taken into consideration, that electromagnetic radiation is received differently depending on it's wavelength? (2) Are the light from the fire and it's heat one in the same? Because when I'm sitting at a campfire and it starts making my face feel hot... (3) The infrared radiation of a hot object is proportional to the fourth power of T, where T is the centigrade temperature + 273, ...		
Permutation	$Q(s, a; \theta)$ **by DRRN-BiLSTM**	$Q(s, a; \theta)$ **by DRRN-Attention**
(1, 2, 3)	131.3	117.0
(1, 3, 2)	132.1	117.0
(2, 1, 3)	67.1	117.0
(2, 3, 1)	45.6	117.0
(3, 1, 2)	84.5	117.0
(3, 2, 1)	60.4	117.0

Table 4: A Q-value prediction example using DRRN-BiLSTM and DRRN-Attention.

8.4 Discussions

As mentioned in Section 7.1, the datasets that we use have a fewer number of comments than the datasets used by previous work. Table 5 com-

Subreddit	# Cmts (their)	# Cmts (our)
askscience	0.32M	0.15M
askmen	1.06M	0.94M
todayilearned	5.11M	4.65M
worldnews	5.99M	4.28M
nfl	6.12M	5.72M

Table 5: The number of comments in the datasets used by He et al. (2016c) and us.

Subreddit	Reward (their)	Reward (our)
askscience	833.9	643.2
askmen	148.0	184.3
todayilearned	697.9	659.0

Table 6: The cumulative reward achieved by the DRRN-BiLSTM + two-stage Q-learning baseline reported by He et al. (2017) and us.

pares the number of comments in the datasets used by (He et al., 2016c) and us. The experiments in (He et al., 2017) used three of the datasets (askscience, askmen, and todayilearned) from (He et al., 2016c). We compare our results and the results reported in (He et al., 2017) of the DRRN-BiLSTM + two-stage Q-learning baseline on these three datasets in Table 6. On the askscience and todayilearned datasets, the results of our implementation are worse than the results reported by them. Since the number of comments in our askscience dataset is only half of that in (He et al., 2017), the results of our implementation on the askscience dataset are significantly worse than the results reported by them. On the askmen dataset, the number of comments that we use is slightly less than the askmen dataset used by them. However, the results of our implementation on askmen are better than the results reported by them. We suspect that the deleted comments may have low karma scores, which cause the agent to achieve a higher cumulative reward.

9 Conclusion

In this work, we propose a new approach to the task of thread popularity prediction and tracking. In our approach, we propose a new neural network architecture, DRRN-Attention, to approximate the Q-function, which well respect the permutation invariance of the comments in an action. Moreover, our approach employs the greedy procedure to approximate the predicted optimal action, which does not require an additional pre-trained model to generate candidate actions. Empirical studies on real data demonstrate that our approach beats the current state-of-the-art in most of the experimental settings.

Acknowledgments

The work described in this paper was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14208815 of the General Research Fund) and the Ministry of Education of China (D.01.16.00101). We would like to thank Jiani Zhang, Hongyi Zhang, Wang Chen, Yifan Gao, and Xiaotian Yu for their comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2014.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

S. R. K. Branavan, David Silver, and Regina Barzilay. 2011. Learning to win by reading manuals in a monte-carlo framework. In *ACL*, 2011.

S. R. K. Branavan, Luke S. Zettlemoyer, and Regina Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *ACL*, 2010.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. 2017. Ask the right questions: Active question reformulation with reinforcement learning. In *ICLR*, 2018.

Jianshu Chen, Chong Wang, Lin Xiao, Ji He, Lihong Li, and Li Deng. 2017. Q-LDA: uncovering latent patterns in text-based sequential decision processes. In *NIPS*, pages 4984–4993, 2017.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2017. A factored neural network model for characterizing online discussions in vector space. In *EMNLP*, pages 2296–2306, 2017.

Justin Cheng, Lada A. Adamic, P. Alex Dow, Jon M. Kleinberg, and Jure Leskovec. 2014. Can cascades be predicted? In *WWW*, pages 925–936, 2014.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *ACL*, pages 484–495, 2017.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016a. Dual learning for machine translation. In *NIPS*, pages 820–828, 2016.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016b. Deep reinforcement learning with a natural language action space. In *ACL*, Volume 1, 2016.

Ji He, Mari Ostendorf, and Xiaodong He. 2017. Reinforcement learning with external knowledge and two-stage q-functions for predicting popular reddit threads. *CoRR*, abs/1704.06217.

Ji He, Mari Ostendorf, Xiaodong He, Jianshu Chen, Jianfeng Gao, Lihong Li, and Li Deng. 2016c. Deep reinforcement learning with a combinatorial action space for predicting popular reddit threads. In *EMNLP*, pages 1838–1848, 2016.

Liangjie Hong, Ovidiu Dan, and Brian D. Davison. 2011. Predicting popular messages in twitter. In *WWW*, pages 57–58, 2011

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *EMNLP*, pages 1192–1202, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *EMNLP*, pages 2355–2365, 2016.

Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 77–85, 2017.

Reddit. 2017. The best of reddit in 2017. https://redditblog.com/2017/12/19/the-best-of-reddit-in-2017/. Accessed: May 22, 2018.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2016. Continuously learning neural dialogue management. *CoRR*, abs/1606.02689.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS*, pages 2440–2448, 2015.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

Chenhao Tan, Lillian Lee, and Bo Pang. 2014. The effect of wording on message propagation: Topic- and author-controlled natural experiments on twitter. In *ACL*, pages 175–185, 2014.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks. *CoRR*, abs/1710.10903.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In *ICLR*, 2015.

Adam Vogel and Daniel Jurafsky. 2010. Learning to follow navigational directions. In *ACL*, pages 806–814, 2010.

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

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *ICLR*, 2015.

Tae Yano and Noah A Smith. 2010. What's worthy of comment? content and comment volume in political blogs. In *ICWSM*, 2010.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan R. Salakhutdinov, and Alexander J. Smola. 2017. Deep sets. In *NIPS*, pages 3394–3404, 2017.

Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. 2017. Dynamic key-value memory networks for knowledge tracing. In *WWW*, pages 765–774, 2017.

Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *CoRR*, abs/1803.07294.