Enhancing Document-Based Question Answering via Interaction between Question Words and POS Tags

Zhipeng Xie

School of Computer Science
Fudan University, Shanghai, China
xiezp@fudan.edu.cn

Abstract. The document-based question answering is to select the answer from a set of candidate sentence for a given question. Most Existing works focus on the sentence-pair modeling, but ignore the peculiars of question-answer pairs. This paper proposes to model the interaction between question words and POS tags, as a special kind of information that is peculiar to question-answer pairs. Such information is integrated into a neural model for answer selection. Experimental results on DBQA Task have shown that our model has achieved better results, compared with several state-of-the-art systems. In addition, it also achieves the best result on NLPCC 2017 Shared Task on DBQA.

Keywords: Question answering, deep learning, question words, part-of-speech tags

1 Introduction

In document-based question answering, one important subtask is to identify sentences as answers from a given document with respect to a question, which is also called *answer selection*. The main problem is how to extract informative features in order to decide whether a candidate sentence contains the semantic and/or syntactic information required by the question. Traditional work on answer selection usually used human-crafted feature engineering that may exploit linguistic tools or external linguistic knowledge resources [12].

Recently, with the upsurge of deep learning, a variety of neural approaches have been proposed to solve the answer selection problem, which have achieved substantial out-performance compared with the traditional methods. These neural approaches usually work by firstly generating the representations of questions and candidates, and then ranking their semantic similarities. Like traditional methods, these neural approaches also assume that an appropriate answer should have high semantic similarity with the question, and they make their judgement mainly based on this assumption.

In some neural approaches, the representations of questions and answers are generated separately. For example, Yu et al. [14] proposed two simple models,

bag-of-words model and bigram model, where bag-of-words model generated the vector representation of a sentence as the centroid of the embeddings of all words in the sentence, and bigram model used one convolutional layer and one average-pooling layer to generate the vector representation of a sentence. Severyn & Moschitti [6] and Feng et al. [1] presented convolutional neural network architectures for reranking pairs of questions and answers. They used convolutional network to generate intermediate representations of input sentences, and then learned how to calculate their similarities.

Some recent neural approaches introduced attention mechanisms and produced conditional representations of answers and questions, taking interdependence between questions and answers into consideration. Tan et al. [8] proposed an attentive Bi-LSTM reader to leverage a simple one-way attention model that emphasizes a certain part of answer based on the question embedding. Yin et al. [13] and dos Santos et al. [5] used two-way attention mechanisms tailored to convolutional neural network.

Although these existing works have achieved good performance in answer selection, most of them treat the answer selection problem as the sentence-pair modeling, and do not take the information peculiar to question-answer pairs into consideration. However, intuitively, the question words that are peculiar to questions often play important roles in answer selection. For example:

- The question word "when" often requires that the answer sentence should contain a time noun;
- the question word "where" requires that the answer sentence should contain a location noun.

Based on this observation, this paper proposes to make use of a special kind of such information, i.e. the interaction between question words and POS tags, and integrates it into a neural model. We evaluate the neural model on the DBQA Shared Task of NLPCC2017 and report the results.

2 The Proposed Model

The document-based question answering task is described as follows. Each question $\mathbf{q}^i \in \mathcal{Q}$ is associcated with a set of candidate sentences, $\mathcal{D}^i = \{\mathbf{d}_1^i, \mathbf{d}_2^i, \ldots, \mathbf{d}_{|\mathcal{D}^i|}^i\}$, where $|\mathbf{d}_j^i|$ is the j-th candidate sentence for the question $\mathbf{q}$. Each candidate sentence $\mathbf{d}_j^i$ is associated with a class label y_j^i, whose value is 1 if the candidate $\mathbf{d}_j^i$ is an appropriate answer to the question $\mathbf{q}^i$, and 0 otherwise. The goal is to train a binary classifier: $f(\mathbf{q}^i, \mathbf{d}_j^i) \rightarrow y_j^i$, where f maps a question-candidate pair to its class labels.

For each pair of question $\mathbf{q}$ and candidate $\mathbf{d}$, assume that $\mathbf{q}$ is a sequence of m words, $q_1 q_2 \ldots q_m$, and $\mathbf{d}$ is a sequence of n words, $d_1 d_2 \ldots d_n$.

We use $\mathbf{f}_{\mathbf{d}|\mathbf{q}}^{wo} = f_{\mathbf{d}|\mathbf{q},1}^{wo} f_{\mathbf{d}|\mathbf{q},2}^{wo} \cdots f_{\mathbf{d}|\mathbf{q},n}^{wo}$ to denote the sequence of word-overlap features for $\mathbf{d}$, where j-the element $f_{\mathbf{d}|\mathbf{q},j}^{wo}$ equals 1 if the j-th word d_j in $\mathbf{d}$ appears in the question $\mathbf{q}$, and 0 otherwise. In addition, $\mathbf{f}_{\mathbf{q}|\mathbf{d}}^{wo} = f_{\mathbf{q}|\mathbf{d},1}^{wo} f_{\mathbf{q}|\mathbf{d},2}^{wo} \cdots f_{\mathbf{q}|\mathbf{d},m}^{wo}$ denotes the sequence of word-overlap features for $\mathbf{q}$, in a similar way.

We hypothesize that a candidate sentence **d** that is qualified to be an answer to the question **q** must satisfy at least the following conditions:

- (Semantic matching) **d** is semantically related to **q**; or in other words, the feature vector of **q** should share high similarity with the feature vector of **d**, in some perspectives;
- (Interaction between question words and POS tags) **d** contains a word whose part-of-speech tag is closely related to the question word of **q**, and the word does not occur in **q**. For example, if the question word "*when*" appears in the question, it is expected that there is a temporal noun in the answer; if the question word "*where*" appears in the question, a sentence that contains a location noun or geographical name is more likely to serve as the answer; if there is the question phrase "*how many*" in the question, the answer will have a large probablity to include a number or quantity.

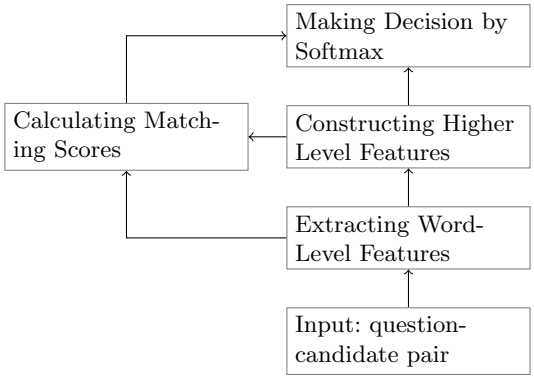

Fig. 1. The architecture of our neural model.

To materialize our hypothesis, we propose a neural model in this paper, with the modular architecture shown in Figure 1. The details are described in the remaining part of this section:

- Section 2.1 will describe the word level features and how to extract them;
- Section 2.2 will introduce how the convolutional and pooling module is used to construct higher-level features (called the intermediate representations) from the lower word-level features;
- Section 2.3 will present four different matching scores and their calculation;
- Section 2.4 will use a hidden layer and a softmax layer to make the decision based on the calculated matching scores and the intermediate representations of the question and the candidate sentence.

2.1 Word-Level Feature Extraction

Before we delve into the details of the neural network, let us first have a look at the word-level features to be used, which include word embeddings, word-overlap feature embeddings, question-word embeddings, POS tag embeddings, and the IDF features. These features are then concatenated to form the vector representations for the words in the question $\mathbf{q}$ and the candidate sentence $\mathbf{d}$.

Word Embeddings Each word w, no matter in the question $\mathbf{q}$ or in the candidate sentence $\mathbf{d}$, can be transformed into a d_{word}-dimensional vector $\mathbf{e}_{word}(w)$ by looking up a word embedding matrix $\mathbf{M}_{word} \in \mathbb{R}^{d_{word} \times |\mathbf{\Sigma}|}$, where $|\mathbf{\Sigma}_{word}|$ denote the size of the word dictionary $\mathbf{\Sigma}_{word}$.

We obtain the word embedding matrix $\mathbf{M}_{word}$ by training the skip-gram neural language model (provided in Word2Vec[1] [4]) on a large unsupervised text corpus of size about 20GB, crawled from Internet . We choose the dimensionality of word embeddings to be 300, window size to be 3, negative sampling to be 5, and number of epochs to be 5. The pretrained word embeddings are kept fixed during the training process.

Word-Overlap Feature Embeddings Let f_j^{wo} denote the word-overlap feature of the j-th word in the candidate sentence $\mathbf{d}$. It can be transformed into a d_{wo}-dimensional vector $\mathbf{e}_{wo}(f_j^{wo})$ by looking up an embedding matrix $\mathbf{M}_{wo}$ of size $2 \times d_{wo}$. Similar to $\mathbf{M}_{qw}$ and $\mathbf{M}_{tag}$, the matrix $\mathbf{M}_{wo}$ is also randomly initialized, and gets tuned in the training process.

To model the interactions between the question words and POS tags, we have to identify all the interested question words from the question $\mathbf{q}$ and the POS tags from the candidate sentence $\mathbf{d}$.

- Firstly, we manually construct a dictionary of question words, denoted by Σ_{qw}. For a given question $\mathbf{q}$, we use $\gamma(\mathbf{q})$ to denote the set of all its words that belong to the question word dictionary $\mathbf{\Sigma}_{qw}$.
- Secondly, we make use of the part-of-speech tagger provided in PyLTP[2] to predict the POS tags for the words in a given candidate sentence $\mathbf{d}$. The POS tag dictionary is the POS tag set used by PyLTP, which contains 28 different POS tags as described at `http://www.ltp-cloud.com/intro/#pos_how`.

Question-Words and Their Embeddings. In a given question $\mathbf{q}$, each question word $q \in \gamma(\mathbf{q})$ is mapped to a d_{qw}-dimensional vector $\mathbf{e}_{qw}(q)$ by looking-up an embedding matrix $\mathbf{M}_{qw}$ of size $|\mathbf{\Sigma}_{qw}| \times d_{qw}$.

[1] http://code.google.com/archive/p/word2vec.
[2] https://github.com/HIT-SCIR/pyltp

POS Tags and Their Embeddings. Let t_j denote the part-of-speech (POS) tag of the j-th word in the candidate sentence $\mathbf{d}$. The POS-tag sequence of $\mathbf{d}$ is then $t_1 t_2 \dots t_n$. Each POS tag t_i $(1 \leq i \leq n)$ is mapped to a d_{tag}-dimensional vector $\mathbf{e}_{tag}(t_i)$ by looking-up an embedding matrix $\mathbf{M}_{tag}$ of size $|\mathbf{\Sigma}_{tag}| \times d_{tag}$.

Both the two embedding matrices $\mathbf{M}_{qw}$ and $\mathbf{M}_{tag}$ are initialized randomly, and get tuned during the training process.

Inverse Document Frequency (IDF). To measure the importance of words, we make use of their inverse document frequencies. We collect all the questions in the training data, and treat each question as a document.

$$idf(w) = \log \frac{|\mathcal{Q}|}{count(w, \mathcal{Q})} \tag{1}$$

where $count(w, \mathcal{Q})$ denotes the number of documents in $\mathcal{Q}$ that contain the word w. It is evident that $idf(w) > 0$ for all words w. The less frequently a word appears in the documents, the higher its IDF value is and the more important the word is.

As to the answer selection task, it is expected that a word in the question with high IDF value has a good match with a word in the answer. Thus, we calculate the IDF value of all the words in the question and feed them into the neural model.

Word-Level Vector Representations Based on the word-level features described above, we can not construct the word-level vector representations for the words in the question $\mathbf{q}$ or the candidate sentence $\mathbf{d}$.

Each word q_i in the question $\mathbf{q}$ is represented as a vector $\mathbf{e}(q_i)$ by concatenating its word embedding, word-overlap feature embedding, question-word embedding, and IDF feature:

$$\mathbf{e}(q_i) = [\mathbf{e}_{word}(q_i); \mathbf{e}_{wo}(q_i); \mathbf{e}_{qw}(q_i); idf(q_i)] \tag{2}$$

The dimensionality of $\mathbf{e}(q_i)$ is $d_{w\_in\_q} = d_{word} + d_{wo} + d_{qw} + 1$.

Each word d_i in the candidate sentence $\mathbf{d}$ is represented as a vector $\mathbf{e}(d_i)$ by concatenating its word embedding, its word-overlap feature embedding, and its POS-tag embedding:

$$\mathbf{e}(d_i) = [\mathbf{e}_{word}(d_i); \mathbf{e}_{wo}(d_i); \mathbf{e}_{tag}(d_i)] \tag{3}$$

Thus, the dimensionality of $\mathbf{e}(d_i)$ is $d_{w\_in\_d} = d_{word} + d_{wo} + d_{tag}$.

2.2 Convolutional and Pooling Module

After each word token has been represented as a vector, the convolutional layer can be applied to compose them in order to extract features at a higher level, and the pooling layer can be used to aggregate the information and reduce the representation.

Convolution Layer. The aim of the convolutional layer is to extract informative higher-level features by composing lower-level ones. Given an input sentence matrix $\mathbf{S} \in \mathbb{R}^{d \times s}$ where s is the length of the sentence and d is the dimensionality of the vector representation of the words, a convolution filter with size f is a matrix of weights: $\mathbf{F} \in \mathbb{R}^{d \times f}$. The convolution operator between $\mathbf{S}$ and $\mathbf{F}$ will result in a vector $\mathbf{c} \in \mathbb{R}^{s+f-1}$, where each component is calculated as follows:

$$\mathbf{c}_i = \sum_{j=1}^{d} \sum_{k=1}^{f} \mathbf{S}[j, i-k+1] \cdot \mathbf{F}[j, f-k+1] \tag{4}$$

Note that in real implementation, each $\mathbf{c}_i$ is added with a bias and then passed through a tanh nonlinear transformation. If there are d_f filters in the convolutional layer, then the output will be a matrix $\mathbf{C} \in \mathbb{R}^{d_f \times (s+f-1)}$.

Pooling. The output from the convolutional layer are passed to the pooling layer, which can aggregate the information and represent a variable-length sentence (question or candidate) as a fixed-sized vector. There are two common choices for pooling functions: max-pooling and average-pooling. We choose to use global max-pooling in the model. It takes maximum value along the temporal dimension of the output matrix $\mathbf{C}$ from the convolutional layer, and results in a vector $\mathbf{v}_{pool} \in \mathbb{R}^{d_f}$ whose i-th component is calculated as:

$$\mathbf{v}_{pool}[i] = \max_{1 \le j \le (s+f-1)} \mathbf{C}[i, j] \tag{5}$$

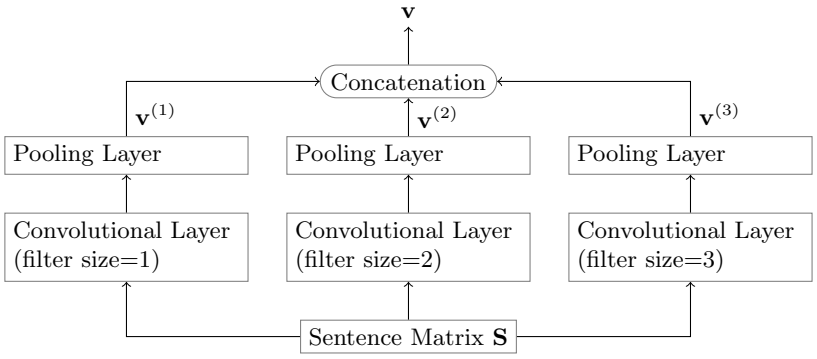

Fig. 2. The convolutional and pooling module

In our model, we use two convolutional and pooling modules, one for questions and the other for candidate sentences. Both the modules share the same architecture, but have different parameters. As illustrated in Figure 2, the module have three convolutional layers whose filter sizes are 1, 2 and 3 respectively.

Each convolutional layer is fed directly with a sentence matrix $\mathbf{S}$ and followed by a global max-pooling layer. The output vectors of the three pooling layers are denoted as $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}$ and $\mathbf{v}^{(3)}$ respectively, which are concatenated into the output vector of the module, $\mathbf{v} = \left[\mathbf{v}^{(1)}; \mathbf{v}^{(2)}; \mathbf{v}^{(3)}\right] \in \mathbb{R}^{3 \times d_f}$.

For a given question $\mathbf{q}$, we use $\mathbf{x}_q$ to denote the output vector of the convolutional and pooling module for questions; while for candidate sentence $\mathbf{d}$, we use annotation $\mathbf{x}_d$. We call $\mathbf{x}_q$ and $\mathbf{x}_d$ the intermediate representations of the question $\mathbf{q}$ and the candidate sentence $\mathbf{d}$ respectively.

2.3 Matching Scores between Questions and Candidate Sentences

Based on the question-candidate pairs, the word-level features and the output from the convolutional and pooling module, we can calculate four matching scores between questions and candidate sentences:

- The first matching score is used to measure the interaction between question words in $\mathbf{q}$ and the POS tags in $\mathbf{d}$;
- The second matching score is used to measure the semantic similarity between $\mathbf{q}$ and $\mathbf{d}$ according to their outputs from the convolutional and pooling modules;
- The last two matching score is the simplest, which measures the word-overlapping and the weighted word-overlapping degrees between $\mathbf{q}$ and $\mathbf{d}$.

Interaction between Question Words and POS Tags. Given a question-candidate pair $(\mathbf{q}, \mathbf{d})$, the interaction score between a question word $q \in \gamma(\mathbf{q})$ and a POS tag t_i is defined as follows:

$$IScore(q, t_i) = \mathbf{e}_{qw}(q)^T \cdot \mathbf{e}_{tag}(t_i) \cdot (1 - f_i^{wo}) \tag{6}$$

A POS tag t_i has a large interaction score with a question word $q \in \gamma(\mathbf{q})$ only if it satisfies two conditions: 1) the tag embedding of t_i has a large inner product with the question-word embedding of q, and 2) the word d_i does not appear in the question $\mathbf{q}$.

For a candidate sentence $\mathbf{d}$ to be an appropriate answer to the question $\mathbf{q}$, it is expected that, for each question word $q \in \gamma(\mathbf{q})$, there exists a word t_i in $\mathbf{d}$ that best match it, i.e., the word t_i has a large interaction score with q. Therefore, the interaction score between $\mathbf{q}$ and $\mathbf{d}$ is defined as:

$$Score_{int}(\mathbf{q}, \mathbf{d}) = \min_{q \in \gamma(\mathbf{q})} \max_{1 \le i \le n} IScore(q, t_i) \tag{7}$$

Semantic Matching between Question and Candidate Sentence Let $\mathbf{x}_q$ and $\mathbf{x}_d$ denote the resulting vector representations produced by the convolutional and pooling modules, on the question $\mathbf{q}$ and the candidate sentence $\mathbf{d}$ respectively. We define the semantic matching score between $\mathbf{q}$ and $\mathbf{d}$ as follow:

$$Score_{sem}(\mathbf{q}, \mathbf{d}) = \mathbf{x}_q^T \mathbf{M}_{match} \mathbf{x}_d \tag{8}$$

where $\mathbf{M}_{match} \in \mathbb{R}^{d_{sent} \times d_{sent}}$ is a similarity matrix (d_{sent} is the dimensionality of $\mathbf{x}_q$ and $\mathbf{x}_d$. The matrix $\mathbf{M}_{match}$ is initialized randomly and gets optimized during the training process.

Word Overlap Score Intuitively, the candidate sentence that has more overlapped words with the question is more likely to be topic-related or semantically-related.

$$Score_{wo}(\mathbf{q}, \mathbf{d}) = \sum_{q \in \mathbf{q}} \mathbb{1}_{\mathbf{d}}(q) \tag{9}$$

where the symbol $\mathbb{1}$ denotes an indicator function, that is:

$$\mathbb{1}_{\mathbf{d}}(q) = \begin{cases} 1, \text{ if } q \in \mathbf{d} \\ 0, \text{ otherwise} \end{cases} \tag{10}$$

In addition, we also calcuate the IDF-weighted word overlap score:

$$Score_{idf}(\mathbf{q}, \mathbf{d}) = \sum_{q \in \mathbf{q}} \mathbb{1}_{\mathbf{d}}(q) \cdot idf(q) \tag{11}$$

where $idf(\cdot)$ is calculated according to Equation (1).

2.4 Softmax Output Module

Given a question-candidate pair $(\mathbf{q}, \mathbf{d})$, it is now represented, by concatenating their intermediate representations ($\mathbf{x}_q$ and $\mathbf{x}_d$) and the four matching scores between them, as a vector $\mathbf{x}_{pair}$ of dimensionality $d_{pair} = 6 \times d_f + 4$, called the final pair representation.

The softmax output module consists of a hidden layer followed by a softmax layer. The hidden layer with d_{hid} hidden units does the following transformation:

$$\mathbf{x}_{hid} = \tanh(\mathbf{W}_{hid} \cdot \mathbf{x}_{pair} + \mathbf{b}_{hid}) \tag{12}$$

where $\mathbf{W}_{hid}$ is a weight matrix of size $d_{hid} \times d_{pair}$, and $\mathbf{b}_{hid}$ is a bias vector of size d_{hid}.

In turn, the vector $\mathbf{x}_{hid}$ is fed to the softmax layer. It first calculate a 2-dimensional score vector $\mathbf{x}_{output} = [x_0, x_1]$:

$$\mathbf{x}_{output} = \mathbf{W}_{output} \cdot \mathbf{x}_{hid} + \mathbf{b}_{output} \tag{13}$$

where $\mathbf{W}_{output} \in \mathbb{R}^{2 \times d_{hid}}$ is a weight matrix, and $\mathbf{b}_{output} \in \mathbb{R}^2$ is a bias vector. Next, the softmax layer applies softmax tranformation on the score vector, resulting in a probability distribution $\mathbf{o} = [o_0, o_1]$ whose component o_i ($i \in 0, 1$) is:

$$o_i = \frac{\exp x_i}{\sum_{j \in \{0,1\}} \exp x_j} \tag{14}$$

2.5 Cross-Entropy Loss Function

Since the answer selection task is formulated as a binary classification problem, we train the model to minimize the cross-entropy loss function defined as follows:

$$L = -\frac{1}{N}\sum_{i=1}^{N} t_i \log o_{i,t_i} \tag{15}$$

where t_i is the golden class label for the i-th pair of question and candidate, and o_{i,t_i} denotes the predicted probability that the i-th pair has the class label t_i.

The parameters are optimized to minimize the cross entropy loss function in Equation (15) by using the Adam algorithm [3], where the learning rate is set to 0.001, beta1 is set to 0.9, and beta2 is set to 0.999.

2.6 Dropout

Dropout is an effective technique to regularize neural networks by randomly drop units during training. It has achieved a great success when working with feed-forward networks [7], convolutional networks, or even recurrent neural networks [15].

In our model, dropout is applied to both the input and the output of the convolutional and pooling modules, with dropout probabilities being 0.4 and 0.7 respectively.

3 Experiments

We evaluate our model on DBQA Shared Task of NLPCC2017. The characteristics of the dataset is described in **Table 1**.

Table 1. Dataset of NLPCC 2017 Shared Task on DBQA

Dataset	Questions	QA-pairs
train	8772	181882
validation	5997	122532

The quality of our DBQA system is evaluated by mean reciprocal rank (**MRR**) and mean average precision (**MAP**) defined as:

– Mean Reciprocal Rank (MRR):

$$MRR = \frac{1}{|\mathcal{Q}|}\sum_{i=1}^{|\mathcal{Q}|}\frac{1}{rank_i} \tag{16}$$

where $|\mathcal{Q}|$ is the total number of questions in the evaluation set, $rank_i$ denotes the position of the first correct answer in the set of candidate sentences for the i-th question.

– Mean Average Precision (MAP):

$$MAP = \frac{1}{|\mathcal{Q}|} \sum_{i=1}^{|\mathcal{Q}|} AveP(C_i, A_i) \tag{17}$$

Here, $AveP(C, A) = \frac{\sum_{k=1}^{n}(P(k) \cdot rel(k))}{min(m,n)}$ denotes the average precision, where k is the rank in the sequence of retrieved candidate sentences, m is the number of correct answer sentences, n is the number of retrieved answer sentences, $P(k)$ is the precision at cut-off k in the list, and $rel(k)$ is an indicator function equaling 1 if the item at rank k is an answer sentence (0 otherwise).

3.1 Performance on the validation dataset

The experimental results of our model are listed in **Table 2**. It is compared with several state-of-the-art systems that participate in the DBQA task of previous year (NLPCC2016).

Table 2. Experimental results on the validation dataset for NLPCC2017 DBQA Task

Model	MRR	MAP
Fu et al. [2]	0.8592	0.8586
Wu et al. [11]	0.8120	0.8111
Wang et al. [9]	0.8008	0.8005
Our model	0.8768	0.8763

In **Table 2**, the model proposed by Fu et al. [2] learns to map the input sentence pairs to vectors, and then to compute their similarity, which achieved the highest MRR and MAP on the DBQA task of NLPCC2016. Wu et al. [11] proposed a hybrid approach to select answer sentences by combining existing model via the rank SVM model, which achieved the third place on the DBQA task of NLPCC2016. Wang et al. [10] integrated count-based features and embedding-based features together, which also achieved a good performance.

3.2 Performance on the test dataset

Table 3. Experimental results on the open test data for NLPCC2017 DBQA Task

	MRR	MAP
Our model	0.7202	0.7166

In addition to the train dataset and the validation dataset, the DBQA task of NLPCC2017 also releases a test dataset where the golden answer annotations are not provided. We obtain our submission result file as follows:

– A larger training dataset is obtained by merging the training and validation datasets.
– On the larger training dataset, our model is trained for 20 epochs, and the model at the final epoch is chosen to make predictions for the test dataset.

The result file by our model has achieved the highest MRR and MAP scores, as listed in **Table 3**, among all the submitted result files.

4 Conclusion

In this paper, we propose to model the interaction between question words and POS tags, as a special kind of information peculiar to question-answer pairs. Such information gets integrated into a neural model to measure the matching degree between questions and answer candidates. Experimental results on NLPCC2017 DBQA task have shown that this neural model has achieve better performance when compared with several state-of-the-art systems.

Acknowledgments

This work is partially supported by National High-Tech R&D Program of China (863 Program) (No. 2015AA015404), and Science and Technology Commission of Shanghai Municipality (No. 14511106802). We are grateful to the anonymous reviewers for their valuable comments.

References

1. Feng, M., Xiang, B., Glass, M.R., Wang, L., Zhou, B.: Applying deep learning to answer selection: A study and an open task. In: Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on. pp. 813–820. IEEE (2015)
2. Fu, J., Qiu, X., Huang, X.: Convolutional deep neural networks for document-based question answering. In: International Conference on Computer Processing of Oriental Languages. pp. 790–797. Springer (2016)
3. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
4. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems (NIPS). pp. 3111–3119 (2013)
5. dos Santos, C.N., Tan, M., Xiang, B., Zhou, B.: Attentive pooling networks. CoRR, abs/1602.03609 (2016)
6. Severyn, A., Moschitti, A.: Learning to rank short text pairs with convolutional deep neural networks. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 373–382 (2015)
7. Srivastava, N.: Improving neural networks with dropout. Ph.D. thesis, University of Toronto (2013)
8. Tan, M., Santos, C.d., Xiang, B., Zhou, B.: Lstm-based deep learning models for non-factoid answer selection. arXiv preprint arXiv:1511.04108 (2015)

9. Wang, B., Niu, J., Ma, L., Zhang, Y., Zhang, L., Li, J., Zhang, P., Song, D.: A chinese question answering approach integrating count-based and embedding-based features. In: Natural Language Understanding and Intelligent Applications - 5th CCF Conference on Natural Language Processing and Chinese Computing, NLPCC 2016, and 24th International Conference on Computer Processing of Oriental Languages, ICCPOL 2016, Kunming, China, December 2-6, 2016, Proceedings. pp. 934–941 (2016)
10. Wang, D., Nyberg, E.: A long short-term memory model for answer sentence selection in question answering. In: ACL (2). pp. 707–712 (2015)
11. Wu, F., Yang, M., Zhao, T., Han, Z., Zheng, D., Zhao, S.: A hybrid approach to DBQA. In: International Conference on Computer Processing of Oriental Languages. pp. 926–933. Springer (2016)
12. Yih, W., Chang, M., Meek, C., Pastusiak, A.: Question answering using enhanced lexical semantic models. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers. pp. 1744–1753 (2013)
13. Yin, W., Schütze, H., Xiang, B., Zhou, B.: Abcnn: Attention-based convolutional neural network for modeling sentence pairs. arXiv preprint arXiv:1512.05193 (2015)
14. Yu, L., Hermann, K.M., Blunsom, P., Pulman, S.: Deep Learning for Answer Sentence Selection. In: NIPS Deep Learning Workshop (Dec 2014), `http://arxiv.org/abs/1412.1632`
15. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)