Question Answering over Freebase with Multi-Column Convolutional Neural Networks

Li Dong[†*] **Furu Wei**[‡] **Ming Zhou**[‡] **Ke Xu**[†]
[†]SKLSDE Lab, Beihang University, Beijing, China
[‡]Microsoft Research, Beijing, China
donglixp@gmail.com {fuwei,mingzhou}@microsoft.com
kexu@nlsde.buaa.edu.cn

Abstract

Answering natural language questions over a knowledge base is an important and challenging task. Most of existing systems typically rely on hand-crafted features and rules to conduct question understanding and/or answer ranking. In this paper, we introduce multi-column convolutional neural networks (MCCNNs) to understand questions from three different aspects (namely, answer path, answer context, and answer type) and learn their distributed representations. Meanwhile, we jointly learn low-dimensional embeddings of entities and relations in the knowledge base. Question-answer pairs are used to train the model to rank candidate answers. We also leverage question paraphrases to train the column networks in a multi-task learning manner. We use FREEBASE as the knowledge base and conduct extensive experiments on the WEBQUESTIONS dataset. Experimental results show that our method achieves better or comparable performance compared with baseline systems. In addition, we develop a method to compute the salience scores of question words in different column networks. The results help us intuitively understand what MCCNNs learn.

1 Introduction

Automatic question answering systems return the direct and exact answers to natural language questions. In recent years, the development of large-scale knowledge bases, such as FREEBASE (Bollacker et al., 2008), provides a rich resource to answer open-domain questions. However, how to understand questions and bridge the gap between natural languages and structured semantics of knowledge bases is still very challenging.

Up to now, there are two mainstream methods for this task. The first one is based on semantic parsing (Berant et al., 2013; Berant and Liang, 2014) and the other relies on information extraction over the structured knowledge base (Yao and Van Durme, 2014; Bordes et al., 2014a; Bordes et al., 2014b). The semantic parsers learn to understand natural language questions by converting them into logical forms. Then, the parse results are used to generate structured queries to search knowledge bases and obtain the answers. Recent works mainly focus on using question-answer pairs, instead of annotated logical forms of questions, as weak training signals (Liang et al., 2011; Krishnamurthy and Mitchell, 2012) to reduce annotation costs. However, some of them still assume a fixed and pre-defined set of lexical triggers which limit their domains and scalability capability. In addition, they need to manually design features for semantic parsers. The second approach uses information extraction techniques for open question answering. These methods retrieve a set of candidate answers from the knowledge base, and the extract features for the question and these candidates to rank them. However, the method proposed by Yao and Van Durme (2014) relies on rules and dependency parse results to extract hand-crafted features for questions. Moreover, some methods (Bordes et al., 2014a; Bordes et al., 2014b) use the summation of question word embeddings to represent questions, which ignores word order information and cannot process complicated questions.

In this paper, we introduce the multi-column convolutional neural networks (MCCNNs) to automatically analyze questions from multiple aspects. Specifically, the model shares the same word embeddings to represent question words.

[*]Contribution during internship at Microsoft Research.

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 260–269,
Beijing, China, July 26-31, 2015.

MCCNNs use different column networks to extract answer types, relations, and context information from the input questions. The entities and relations in the knowledge base (namely FREEBASE in our experiments) are also represented as low-dimensional vectors. Then, a score layer is employed to rank candidate answers according to the representations of questions and candidate answers. The proposed information extraction based method utilizes question-answer pairs to automatically learn the model without relying on manually annotated logical forms and hand-crafted features. We also do not use any pre-defined lexical triggers and rules. In addition, the question paraphrases are also used to train networks and generalize for the unseen words in a multi-task learning manner. We have conducted extensive experiments on WEBQUESTIONS. Experimental results illustrate that our method outperforms several baseline systems.

The contributions of this paper are three-fold:

- We introduce multi-column convolutional neural networks for question understanding without relying on hand-crafted features and rules, and use question paraphrases to train the column networks and word vectors in a multi-task learning manner;
- We jointly learn low-dimensional embeddings for the entities and relations in FREEBASE with question-answer pairs as supervision signals;
- We conduct extensive experiments on the WEBQUESTIONS dataset, and provide some intuitive interpretations for MCCNNs by developing a method to detect salient question words in the different column networks.

2 Related Work

The state-of-the-art methods for question answering over a knowledge base can be classified into two classes, i.e., semantic parsing based and information retrieval based.

Semantic parsing based approaches aim at learning semantic parsers which parse natural language questions into logical forms and then query knowledge base to lookup answers. The most important step is mapping questions into predefined logical forms, such as combinatory categorial grammar (Cai and Yates, 2013) and dependency-based compositional semantics (Liang et al., 2011). Some semantic parsing based systems required manually annotated logical forms to train the parsers (Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2010). These annotations are relatively expensive. So recent works (Liang et al., 2011; Kwiatkowski et al., 2013; Berant et al., 2013; Berant and Liang, 2014; Bao et al., 2014; Reddy et al., 2014) mainly aimed at using weak supervision (question-answer pairs) to effectively train semantic parsers. These methods achieved comparable results without using logical forms annotated by experts. However, some methods relied on lexical triggers or manually defined features.

On the other hand, information retrieval based systems retrieve a set of candidate answers and then conduct further analysis to obtain answers. Their main difference is how to select correct answers from the candidate set. Yao and Van Durme (2014) used rules to extract question features from dependency parse of questions, and used relations and properties in the retrieved topic graph as knowledge base features. Then, the production of these two kinds of features was fed into a logistic regression model to classify the question's candidate answers into correct/wrong. In contrast, we do not use rules, dependency parse results, or hand-crafted features for question understanding. Some other works (Bordes et al., 2014a; Bordes et al., 2014b) learned low-dimensional vectors for question words and knowledge base constitutes, and used the sum of vectors to represent questions and candidate answers. However, simple vector addition ignores word order information and high-order n-grams. For example, the question representations of *who killed A* and *who A killed* are same in the vector addition model. We instead use multi-column convolutional neural networks which are more powerful to process complicated question patterns. Moreover, our multi-column network architecture distinguishes between information of answer type, answer path and answer context by learning multiple column networks, while the addition model mixes them together.

Another line of related work is applying deep learning techniques for the question answering task. Grefenstette et al. (2014) proposed a deep architecture to learn a semantic parser from annotated logic forms of questions. Iyyer et al. (2014) introduced dependency-tree recursive neural networks for the quiz bowl game which asked players to answer an entity for a given paragraph. Yu et

al. (2014) proposed a bigram model based on convolutional neural networks to select answer sentences from text data. The model learned a similarity function between questions and answer sentences. Yih et al. (2014) used convolutional neural networks to answer single-relation questions on REVERB (Fader et al., 2011). However, the system worked on relation-entity triples instead of more structured knowledge bases. For instance, the question shown in Figure 1 is answered by using several triples in FREEBASE. Also, we can utilize richer information (such as entity types) in structured knowledge bases.

3 Setup

Given a natural language question $q = w_1 \dots w_n$, we retrieve related entities and properties from FREEBASE and use them as the candidate answers C_q. Our goal is to score these candidates and predict answers. For instance, the correct output of the question *when did Avatar release in UK* is *2009-12-17*. It should be noted that there may be several correct answers for a question. In order to train the model, we use question-answer pairs without annotated logic forms. We further describe the datasets used in our work as follows:

WebQuestions This dataset (Berant et al., 2013) contains 3,778 training instances and 2,032 test instances. We further split the training instances into the training set and the development set by 80%/20%. The questions were collected by querying the Google Suggest API. A breadth-first search beginning with *wh-* was conducted. Then, answers were annotated in Amazon Mechanical Turk. All the answers can be found in FREEBASE.

Freebase It is a large-scale knowledge base that consists of general facts (Bollacker et al., 2008). These facts are organized as subject-property-object triples. For example, the fact *Avatar is directed by James Cameron* is represented by (/m/0bth54, film.film.directed_by, /m/03_gd) in RDF format. The preprocess method presented in (Bordes et al., 2014a) was used to make FREEBASE fit in memory. Specifically, we kept the triples where one of the entities appeared in the training/development set of WEBQUESTIONS or CLUEWEB extractions provided in (Lin et al., 2012), and removed the entities appearing less than five times. Then, we obtained 18M triples that contained 2.9M entities and 7k relation types. As described in (Bordes et al., 2014a), this preprocess method does not ease the task because WEBQUESTIONS only contains about 2k entities.

WikiAnswers Fader et al. (2013) extracted the similar questions on WIKIANSWERS and used them as question paraphrases. There are 350,000 paraphrase clusters which contain about two million questions. They are used to generalize for unseen words and question patterns.

4 Methods

The overview of our framework is shown in Figure 1. For instance, for the question *when did Avatar release in UK*, the related nodes of the entity *Avatar* are queried from FREEBASE. These related nodes are regarded as candidate answers (C_q). Then, for every candidate answer a, the model predicts a score $S(q, a)$ to determine whether it is a correct answer or not.

We use multi-column convolutional neural networks (MCCNNs) to learn representations of questions. The models share the same word embeddings, and have multiple columns of convolutional neural networks. The number of columns is set to three in our QA task. These columns are used to analyze different aspects of a question, i.e., answer path, answer context, and answer type. The vector representations learned by these columns are denoted as $\mathbf{f}_1(q), \mathbf{f}_2(q), \mathbf{f}_3(q)$. We also learn embeddings for the candidate answers appeared in FREEBASE. For every candidate answer a, we compute its vector representations and denote them as $\mathbf{g}_1(a), \mathbf{g}_2(a), \mathbf{g}_3(a)$. These three vectors correspond to the three aspects used in question understanding. Using these vector representations defined for questions and answers, we can compute the score for the question-answer pair (q, a). Specifically, the scoring function $S(q, a)$ is defined as:

$$S(q,a) = \underbrace{\mathbf{f}_1(q)^{\mathsf{T}}\mathbf{g}_1(a)}_{answer\ path} + \underbrace{\mathbf{f}_2(q)^{\mathsf{T}}\mathbf{g}_2(a)}_{answer\ context} + \underbrace{\mathbf{f}_3(q)^{\mathsf{T}}\mathbf{g}_3(a)}_{answer\ type} \quad (1)$$

where $\mathbf{f}_i(q)$ and $\mathbf{g}_i(a)$ have the same dimension. As shown in Figure 1, the score layer computes scores and adds them together.

4.1 Candidate Generation

The first step is to retrieve candidate answers from FREEBASE for a question. Questions should contain an identified entity that can be linked to the

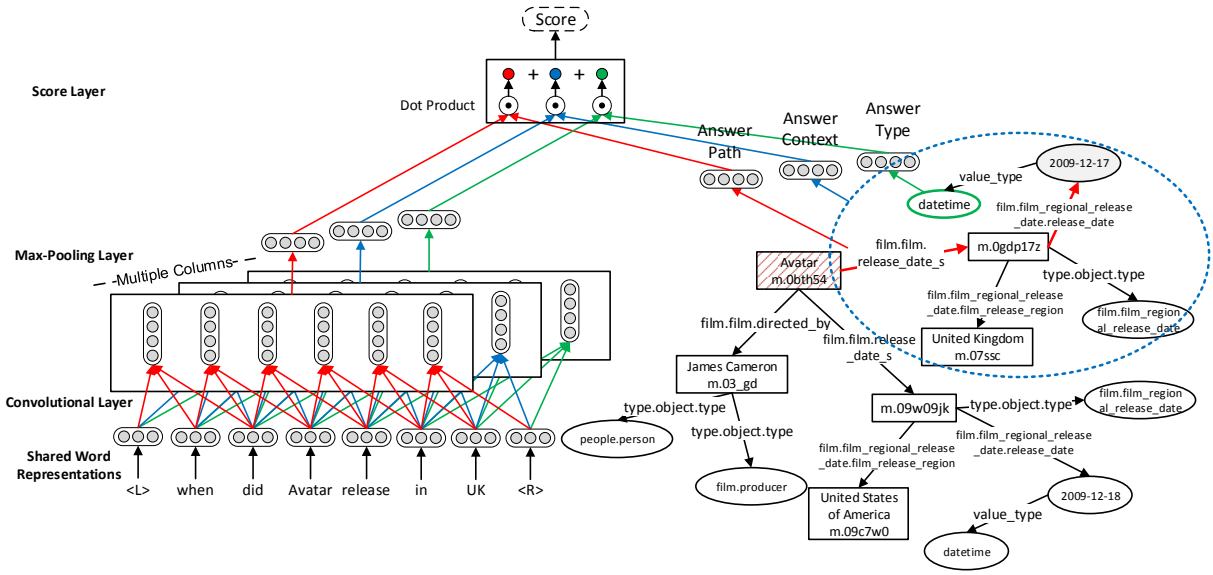

Figure 1: Overview for the question-answer pair *(when did Avatar release in UK, 2009-12-17)*. Left: network architecture for question understanding. Right: embedding candidate answers.

knowledge base. We use the Freebase Search API (Bollacker et al., 2008) to query named entities in a question. If there is not any named entity, noun phrases are queried. We use the top one entity in the ranked list returned by the API. This entity resolution method was also used in (Yao and Van Durme, 2014). Better methods can be developed, while it is not the focus of this paper. Then, all the 2-hops nodes of the linked entity are regarded as the candidate answers. We denote the candidate set for the question q as C_q.

4.2 MCCNNs for Question Understanding

MCCNNs use multiple convolutional neural networks to learn different aspects of questions from shared input word embeddings. For every single column, the network structure presented in (Collobert et al., 2011) is used to tackle the variable-length questions.

We present the model in the left part of Figure 1. Specifically, for the question $q = w_1 \dots w_n$, the lookup layer transforms every word into a vector $\mathbf{w}_j = \mathbf{W}_v \mathbf{u}(w_j)$, where $\mathbf{W}_v \in \mathbb{R}^{d_v \times |V|}$ is the word embedding matrix, $\mathbf{u}(w_j) \in \{0,1\}^{|V|}$ is the one-hot representation of w_j, and $|V|$ is the vocabulary size. The word embeddings are parameters, and are updated in the training process.

Then, the convolutional layer computes representations of the words in sliding windows. For the i-th column of MCCNNs, the convolutional layer computes n vectors for question q. The j-th vector is:

$$\mathbf{x}_j^{(i)} = \mathbf{h}\left(\mathbf{W}^{(i)}\left[\mathbf{w}_{j-s}^{\mathsf{T}} \dots \mathbf{w}_j^{\mathsf{T}} \dots \mathbf{w}_{j+s}^{\mathsf{T}}\right]^{\mathsf{T}} + \mathbf{b}^{(i)}\right) \tag{2}$$

where $(2s+1)$ is the window size, $\mathbf{W}^{(i)} \in \mathbb{R}^{d_q \times (2s+1)d_v}$ is the weight matrix of convolutional layer, $\mathbf{b}^{(i)} \in \mathbb{R}^{d_q \times 1}$ is the bias vector, and $\mathbf{h}(\cdot)$ is the nonlinearity function (such as softsign, tanh, and sigmoid). Paddings are used for left and right absent words.

Finally, a max-pooling layer is followed to obtain the fixed-size vector representations of questions. The max-pooling layer in the i-th column of MCCNNs computes the representation of the question q via:

$$\mathbf{f}_i(q) = \max_{j=1,\dots,n} \{\mathbf{x}_j^{(i)}\} \tag{3}$$

where $\max\{\cdot\}$ is an element-wise operator over vectors.

4.3 Embedding Candidate Answers

Vector representations $\mathbf{g}_1(a), \mathbf{g}_2(a), \mathbf{g}_3(a)$ are learned for the candidate answer a. The vectors are employed to represent different aspects of a. The embedding methods are described as follows:

Answer Path The answer path is the set of relations between the answer node and the entity asked in question. As shown in Figure 1, the 2-hops path between the entity *Avatar* and the correct answer is (film.film.release_date_s,

film.film_regional_release_date.release_date). The vector representation $\mathbf{g}_1(a)$ is computed via $\mathbf{g}_1(a) = \frac{1}{\|\mathbf{u}_p(a)\|_1}\mathbf{W}_p\mathbf{u}_p(a)$, where $\|\cdot\|_1$ is 1-norm, $\mathbf{u}_p(a) \in \mathbb{R}^{|R|\times 1}$ is a binary vector which represents the presence or absence of every relation in the answer path, $\mathbf{W}_p \in \mathbb{R}^{d_q\times|R|}$ is the parameter matrix, and $|R|$ is the number of relations. In other words, the embeddings of relations that appear on the answer path are averaged.

Answer Context The 1-hop entities and relations connected to the answer path are regarded as the answer context. It is used to deal with constraints in questions. For instance, as shown in Figure 1, the release date of Avatar in UK is asked, so it is not enough that only the triples on answer path are considered. With the help of context information, the release date in UK has a higher score than in USA. The context representation is $\mathbf{g}_2(a) = \frac{1}{\|\mathbf{u}_c(a)\|_1}\mathbf{W}_c\mathbf{u}_c(a)$, where $\mathbf{W}_c \in \mathbb{R}^{d_q\times|C|}$ is the parameter matrix, $\mathbf{u}_c(a) \in \mathbb{R}^{|C|\times 1}$ is a binary vector expressing the presence or absence of context nodes, and $|C|$ is the number of entities and relations which appear in answer context.

Answer Type Type information in FREEBASE is an important clue to score candidate answers. As illustrated in Figure 1, the type of *2009-12-17* is datetime, and the type of *James Cameron* is people.person and film.producer. For the example question *when did Avatar release in UK*, the candidate answers whose types are datetime should be assigned with higher scores than others. The vector representation is defined as $\mathbf{g}_3(a) = \frac{1}{\|\mathbf{u}_t(a)\|_1}\mathbf{W}_t\mathbf{u}_t(a)$, where $\mathbf{W}_t \in \mathbb{R}^{d_q\times|T|}$ is the matrix of type embeddings, $\mathbf{u}_t(a) \in \mathbb{R}^{|T|\times 1}$ is a binary vector which indicates the presence or absence of answer types, and $|T|$ is the number of types. In our implementation, we use the relation common.topic.notable_types to query types. If a candidate answer is a property value, we instead use its value type (e.g., float, string, datetime).

4.4 Model Training

For every correct answer $a \in A_q$ of the question q, we randomly sample k wrong answers a' from the set of candidate answers C_q, and use them as negative instances to estimate parameters. To be more specific, the hinge loss is considered for pairs (q, a) and (q, a'):

$$l\left(q, a, a'\right) = \left(m - S(q, a) + S(q, a')\right)_+ \quad (4)$$

where $S(\cdot,\cdot)$ is the scoring function defined in Equation (1), m is the margin parameter employed to regularize the gap between two scores, and $(z)_+ = \max\{0, z\}$. The objective function is:

$$\min \sum_q \frac{1}{|A_q|} \sum_{a\in A_q} \sum_{a'\in R_q} l\left(q, a, a'\right) \quad (5)$$

where $|A_q|$ is the number of correct answers, and $R_q \subseteq C_q \setminus A_q$ is the set of k wrong answers.

The back-propagation algorithm (Rumelhart et al., 1986) is used to train the model. It back-propagates errors from top to the other layers. Derivatives are calculated and gathered to update parameters. The AdaGrad algorithm (Duchi et al., 2011) is then employed to solve this non-convex optimization problem. Moreover, the max-norm regularization (Srebro and Shraibman, 2005; Srivastava et al., 2014) is used for the column vectors of parameter matrices.

4.5 Inference

During the test, we retrieve all the candidate answers C_q for the question q. For every candidate $\hat{a}$, we compute its score $S(q, \hat{a})$. Then, the candidate answers with the highest scores are regarded as predicted results.

Because there may be more than one correct answers for some questions, we need a criterion to determine the score threshold. Specifically, the following equation is used to determine outputs:

$$\hat{A}_q = \{\hat{a} \mid \hat{a} \in C_q \ \ and \ \max_{a'\in C_q}\{S(q, a')\} - S(q, \hat{a}) < m\} \quad (6)$$

where m is the margin defined in Equation (4). The candidates whose scores are not far from the best answer are regarded as predicted results.

Some questions may have a large set of candidate answers. So we use a heuristic method to prune their candidate sets. To be more specific, if the number of candidates on the same answer path is greater than 200, we randomly keep 200 candidates for this path. Then, we score and rank all these generated candidate answers together. If one of the candidates on the pruned path is regarded as a predicted answer, we further score the other candidates that are pruned on this path and determine the final results.

4.6 Question Paraphrases for Multi-Task Learning

We use the question paraphrases dataset WIKIANSWERS to generalize for words and question patterns which are unseen in the training set of question-answer pairs. The question understanding results of paraphrases should be same. Consequently, the representations of two paraphrases computed by the same column of MCCNNs should be similar. We use dot similarity to define the hinge loss $l_p(q_1, q_2, q_3)$ as:

$$l_p(q_1, q_2, q_3) = \sum_{i=1}^{3} \Big(m_p - \mathbf{f}_i(q_1)^\top \mathbf{f}_i(q_2) + \mathbf{f}_i(q_1)^\top \mathbf{f}_i(q_3) \Big)_+ \quad (7)$$

where q_1, q_2 are questions in the same paraphrase cluster P, q_3 is randomly sampled from another cluster, and m_p is the margin. The objective function is defined as:

$$\min \sum_{P} \sum_{q_1, q_2 \in P} \sum_{q_3 \in R_P} l_p(q_1, q_2, q_3) \quad (8)$$

where R_P contains k_p questions which are randomly sampled from other clusters. The same optimization algorithm described in Section 4.4 is used to update parameters.

5 Experiments

In order to evaluate the model, we use the dataset WEBQUESTIONS (Section 3) to conduct experiments.

Settings The development set is used to select hyper-parameters in the experiments. The nonlinearity function $f = \tanh$ is employed. The dimension of word vectors is set to 25. They are initialized by the pre-trained word embeddings provided in (Turian et al., 2010). The window size of MCCNNs is 5. The dimension of the pooling layers and the dimension of answer embeddings are set to 64. The parameters are initialized by the techniques described in (Bengio, 2012). The max value used for max-norm regularization is 3. The initial learning rate used in AdaGrad is set to 0.01. A mini-batch consists of 10 question-answer pairs, and every question-answer pair has k negative samples that are randomly sampled from its candidate set. The margin values in Equation (4) and Equation (7) is set to $m = 0.5$ and $m_p = 0.1$.

Method	F1	P@1
(Berant et al., 2013)	31.4	-
(Berant and Liang, 2014)	39.9	-
(Bao et al., 2014)	37.5	-
(Yao and Van Durme, 2014)	33.0	-
(Bordes et al., 2014a)	39.2	40.4
(Bordes et al., 2014b)	29.7	31.3
MCCNN (our)	**40.8**	**45.1**

Table 1: Evaluation results on the test split of WEBQUESTIONS.

5.1 Experimental Results

The evaluation metrics macro F1 score (Berant et al., 2013) and precision @ 1 (Bordes et al., 2014a) are reported. We use the official evaluation script provided by Berant et al. (2013) to compute the F1 score. Notably, the F1 score defined in (Yao and Van Durme, 2014) is slightly different from others (how to compute scores for the questions without predicted results). We instead use the original definition in experiments.

As shown in Table 1, our method achieves better or comparable results than baseline methods on WEBQUESTIONS. To be more specific, the first three rows are semantic parsing based methods, and the other baselines are information extraction based methods. These approaches except (Bordes et al., 2014a; Bordes et al., 2014b) rely on handcrafted features and predefined rules. The results show that automatically question understanding can be as good as the models using manually designed features. Besides, our multi-column convolutional neural networks based model outperforms the methods that use the sum of word embeddings as question representations (Bordes et al., 2014a; Bordes et al., 2014b).

5.2 Model Analysis

We also conduct ablation experiments to compare the results using different experiment settings. As shown in Table 2, the abbreviation *w/o* means removing a particular part from the model. We find that answer path information is most important among these three columns, and answer type information is more important than answer context information. The reason is that answer path and answer type are more direct clues for questions, but answer context is used to handle additional constraints in questions which are less common in the dataset. Moreover, we compare to the

Setting	F1	P@1
all	**40.8**	**45.1**
w/o path	32.5	37.1
w/o type	37.7	40.9
w/o context	39.1	41.0
w/o multi-column	38.4	41.8
w/o paraphrase	40.0	43.9
1-hop	29.3	32.2

Table 2: Evaluation results of different settings on the test split of WEBQUESTIONS. w/o path/type/context: without using the specific column. w/o multi-column: tying parameters of multiple columns. w/o paraphrase: without using question paraphrases for training. 1-hop: using 1-hop paths to generate candidate answers.

model using single-column networks (w/o multi-column), i.e., tying the parameters of different columns. The results indicate that using multiple columns to understand questions from different aspects improves the performance. Besides, we find that using question paraphrases in a multi-task learning manner contributes to the performance. In addition, we evaluate the results only using 1-hop paths to generate candidate answers. Compared to using 2-hops paths, we find that the performance drops significantly. This indicates only using the nodes directly connected to the queried entity in FREEBASE cannot handle many questions.

5.3 Salient Words Detection

In order to analyze the model, we detect salient words in questions. The salience score of a question word depends on how much the word affects the computation of question representation. In other words, if a word plays more important role in the model, its salience score should be larger.

We compute several salience scores for a same word to illustrate its importance in different columns of networks. For the i-th column, the salience score of word w_j in the question $q = w_1^n$ is defined as:

$$e_i(w_j) = \left\| \mathbf{f}_i\left(w_1^n\right) - \mathbf{f}_i\left(w_1^{j-1} w_j' w_{j+1}^n\right) \right\|_2 \quad (9)$$

where the word w_j is replaced with w_j', and $\|\cdot\|_2$ denotes Euclidean norm. In practice, we replace w_j with several stop words (such as *is*, *to*, and *a*), and then compute their average score.

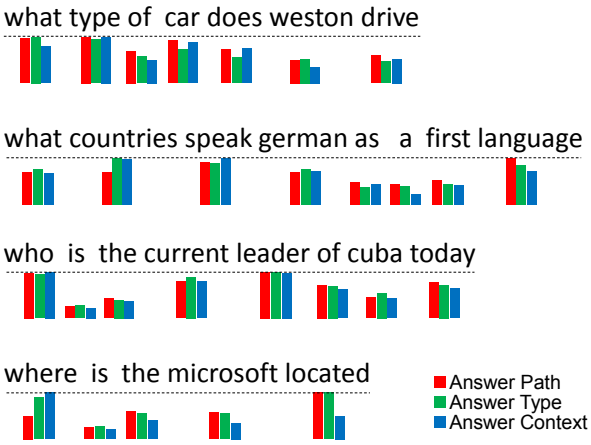

Figure 2: Salient words detection results for questions. From left to right, the three bars of every word correspond to salience scores in answer path column, answer type column, and answer context column, respectively. The salience scores are normalized by the max values of different columns.

As shown in Figure 2, we compute salience scores for several questions, and normalize them by the max values in different columns. We clearly see that these words play different roles in a question. The overall conclusion is that the *wh-* words (such as *what*, *who* and *where*) tend to be important for question understanding. Moreover, nouns dependent of the *wh-* words and verbs are important clues to obtain question representations. For instance, the figure demonstrates that the nouns *type/country/leader* and the verbs *speak/located* are salient in the columns of networks. These observations agree with previous works (Li and Roth, 2002). Some manually defined rules (Yao and Van Durme, 2014) used in the question answering task are also based on them.

5.4 Examples

Question representations computed by different columns of MCCNNs are used to query their most similar neighbors. We use cosine similarity in experiments. This experiment demonstrates whether the model learns different aspects of questions. For example, if a column of networks is employed to analyze answer types, the answer types of nearest questions should be same as the query.

As shown in Table 3, these three columns of table correspond to different columns of networks. To be more specific, the first column is used to process answer path. We find that the model learns different question patterns for the same

Column 1 (Answer Path)	Column 2 (Answer Type)	Column 3 (Answer Context)
what to do in hollywood can this weekend	**where be george washington originally from**	**where do charle draw go to college**
what to do in midland tx this weekend	where be george washington carver from	where do kevin love go to college
what to do in cancun with family	where be george bush from	where do pauley perrette go to college
what to do at fairfield can	where be the thame river source	where do kevin jame go to college
what to see in downtown asheville nc	where be the main headquarters of google	where do charle draw go to high school
what to see in toronto top 10	in what town do ned kelly and he family grow up	where do draw bree go to college wikianswer
who found collegehumor	**who be the leader of north korea today**	**who be judy garland father**
who found the roanoke settlement	who be the leader of syrium now	who be clint eastwood date
who own skywest	who be the leader of cuba 2012	who be emma stone father
who start mary kay	who be the leader of france 2012	who be robin robert father
who be the owner of kfc	who be the current leader of cuba today	who miley cyrus engage to
who own wikimedium foundation	who be the minority leader of the house of representative now	who be chri cooley marry to
what type of money do japanese use	**what be the two official language of paraguay**	**what be the timezone in vancouver**
what kind of money do japanese use	what be the local language of israel	what be my timezone in californium
what type of money do jamaica use	what be the four official language of nigerium	what be los angeles california time zone
what type of currency do brazil use	what be the official language of jamaica	what be my timezone in oklahoma
what type of money do you use in cuba	what be the dominant language of jamaica	what be my timezone in louisiana
what money do japanese use	what be the official language of brazil now	what be the time zone in france

Table 3: Using question representations obtained by different column networks to query the nearest neighbors. From left to right, the three columns are used to analyze information about answer path, answer type, and answer context, respectively. Lemmatization is used to better show question patterns.

path. For instance, the vector representations of "*who found/own/start* *" and "*who be the owner of* *" obtained by the first column are similar. The second column is employed to extract answer type information from questions. The answer types of example questions in Table 3 are same, while they may ask different relations. The third column learns to embed question information into answer context. We find that the similar questions are clustered together by this column.

5.5 Error Analysis

We investigate the predicted results on the development set, and show several error causes as follows.

Candidate Generation Some entity mentions in questions are linked incorrectly, hence we cannot obtain the desired candidate answers. As described in (Yao and Van Durme, 2014), the Freebase Search API returned correct entities for 86.4% of questions in top one results. Because some questions use the abbreviation or a part of its mention to express an entity. For example, it is not trivial to link *jfk* to *John F. Kennedy* in the question "*where did jfk and his wife live*". A better entity retrieval step should be developed for the open question answering scenario.

Time-Aware Questions We need to compare date values for some time-aware questions. For instance, to answer the question "*who is johnny cash's **first** wife*", we have to know the order of several marriages by comparing the marriage date. Its correct response should contain only one entity (*vivian liberto*). However, our system additionally outputs *june carter cash* who is his second wife, because both the candidate answers are connected to *johnny cash* by the relation people.person.spouse_s. In order to solve this issue, we need to define some ad-hoc operators used for comparisons or develop more advanced semantic representations.

Ambiguous Questions Some questions are ambiguous to obtain their correct representations. For example, the question *what has anna kendrick been in* is used to ask what movies she has played in. This question does not have explicit clue words to indicate the meanings, so it is difficult to rank the candidates. Moreover, the question *who is aidan quinn* is employed to ask what his occupation is. It also lacks sufficient clues for question understanding, and using *who is* to ask occupation is rare in the training data.

6 Conclusion and Future Work

This paper presents a method for question answering over FREEBASE using multi-column convolutional neural networks (MCCNNs). MCCNNs share the same word embeddings, and use multiple columns of convolutional neural networks to learn the representations of different aspects of questions. Accordingly, we use low-dimensional embeddings to represent multiple aspects of candidate answers, i.e., answer path, answer type, and answer context. We estimate the parameters from question-answer pairs, and use question paraphrases to train the columns of MCCNNs in a multi-task learning manner. Experimental results on WEBQUESTIONS show that our approach

achieves better or comparable performance comparing with baselines. There are several interesting directions that are worth exploring in the future. For instance, we are integrating more external knowledge source, such as CLUEWEB (Lin et al., 2012), to train MCCNNs in a multi-task learning manner. Furthermore, as our model is capable of detecting the most important words in a question, it would be interesting to use the results to mine effective question patterns.

Acknowledgments

This research was supported by NSFC (Grant No. 61421003) and the fund of the State Key Lab of Software Development Environment (Grant No. SKLSDE-2015ZX-05).

References

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 967–976. Association for Computational Linguistics.

Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425. Association for Computational Linguistics.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544. Association for Computational Linguistics.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *International Conference on Management of Data*, pages 1247–1250.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620. Association for Computational Linguistics.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I*, pages 165–180.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 423–433. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, July.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618. Association for Computational Linguistics.

Edward Grefenstette, Phil Blunsom, Nando de Freitas, and Moritz Karl Hermann, 2014. *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, chapter A Deep Architecture for Semantic Parsing, pages 22–27. Association for Computational Linguistics.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 633–644. Association for Computational Linguistics.

Jayant Krishnamurthy and Tom M Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 754–765. Association for Computational Linguistics.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1223–1233. Association for Computational Linguistics.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556. Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING*, pages 1–7.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, pages 590–599.

Thomas Lin, Mausam, and Oren Etzioni. 2012. Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 84–88, Stroudsburg, PA, USA. Association for Computational Linguistics.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association of Computational Linguistics – Volume 2, Issue 1*, pages 377–392.

D.E. Rumelhart, G.E. Hinton, and R.J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Nathan Srebro and Adi Shraibman. 2005. Rank, trace-norm and max-norm. In *Proceedings of the 18th annual conference on Learning Theory*, pages 545–560. Springer-Verlag.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *ACL*.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966. Association for Computational Linguistics.

Xuchen Yao, Jonathan Berant, and Benjamin Van Durme, 2014. *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, chapter Freebase QA: Information Extraction or Semantic Parsing?, pages 82–86. Association for Computational Linguistics.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 643–648. Association for Computational Linguistics.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. In *NIPS Deep Learning Workshop*, December.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *In Proceedings of the 21st Conference on Uncertainty in AI*, pages 658–666.