Convolutional Neural Network-based Question Answering over Knowledge Base with Type Constraint

Yongrui Chen[1], Huiying Li[1], Zejian Xu[1]

[1] School of Computer Science and Engineering, Southeast University, Nanjing, China
18795865715@163.com
{huiyingli, 220161578}@seu.edu.cn

Abstract. We propose a staged framework for question answering over a large-scale structured knowledge base. Following existing methods based on semantic parsing, our method relies on various components for solving different sub-tasks of the problem. In the first stage, we directly use the result of entity linking to obtain the topic entity in a question, and simplify the process as a semantic matching problem. We train a neural network to match questions and predicate sequences to get a rough set of candidate answer entities from the knowledge base. Unlike traditional methods, we also consider entity type as a constraint on candidate answers to remove wrong candidates from the rough set in the second stage. By applying a convolutional neural network model to match questions and predicate sequences and a type constraint to filter candidate answers, our method achieves an average F_1 measure of 74.8% on the WEBQUESTIONSSP dataset, it is competitive with state-of-the-art semantic parsing approaches.

Keywords: Question Answering, Type Constraint, Convolutional Neural Network, Knowledge Base, Semantic Parsing.

1 Introduction

Large-scale knowledge base (KB) which is a structured database comprising vast amounts of world's facts, such as Freebase [1], DBpedia [2], have become an important resource for open-domain question answering (QA). Question answering over knowledge bases (KB-QA) is an active research area where the goal is to provide crisp answers to natural language questions [3]. One direction in KB-QA performs the answering via semantic parsing: translating natural language questions posed by humans into structured queries (e.g. SPARQL). However, the translating from question to the corresponding KB query is still a challenging task. One challenge is the *lexical gap* [4] that the relation expressed in a question can be quite different from which used in the KB. Another challenge is the *additional constraints* that need to be handled when dealing with complex questions. In this paper, we handle these problems with a staged framework.

Our approach has two stages. In the first stage, after obtaining the *topic entity* of the question, we detect the relation asked by the question for all the relations (relation or relation chain with limited length) that starts from the topic entity. The candidate answer entities can be retrieved with the topic entity and the detected relation. In the second stage, we add a *type constraint* to the candidate answer entities. We predict the answer entity types to remove wrong answers and obtain more accurate final answers. Inspired by [5], we employ the neural network models to handle the relation detection and type prediction in our approach.

It is a significant difference from previous methods that we consider the constraint of the entity type. We notice that a wrong answer usually has an incorrect type. For example, given question "What highschool did Harper Lee go to?", five candidate answer entities can be retrieved along the relation chain "education-institution" in the first stage of our approach. However, only the entity "Monroe County High School" with type "School" is the correct answer, other candidates' type "College/University" are wrong. For improving the accuracy of results, we add type constraint on the candidate answer entities. Our question answering system improves the state-of-the-art result of on the WEBQUESTIONSSP dataset [6] to 74.8% in F1, a 3.1% absolute gain compared to the best existing method.

The rest of this paper is structured as follows. Sec. 2 introduces the overview of related work for KB-QA. Sec. 3 presents our staged approach based on CNN. The experimental results are shown in Sec. 4. Finally, Sec. 5 concludes the paper.

2 Related Work

An important line of research for KB-QA constructs an end-to-end system by deep learning powered similarity matching. The aim of these approaches is to learn semantic representations of both the question and the KB elements with an encoder-compare framework, so that the correct KB element is the nearest neighbor of the question in the learned vector space [7]. These approaches are different on the used neural network models, such as CNN or RNN, and the input granularity, such as word-level or character-level. Denis et al. [4] concentrate two granularities of relation. They propose a GRU-based model on both word and character level. Yu et al. [8] propose a hierarchical RNN enhanced by residual learning, which detects different levels of abstraction. Qu et al. [9] propose an attentive RNNs with similarity matrix based CNNs to preserve more original words interaction information. These end-to-end methods usually have only a single process in order to avoid complex NLP pipeline constructions and error propagation. Another advantage is they can be retrained or reused for a different domain [4]. However, the obvious drawback of these methods is the results are often unexplainable.

Another important line of research in this domain performs answering based on semantic parsing. The core idea of semantic parsing is to translate a question into a structured KB query. Then, the answer to the question can be retrieved simply by executing the query. Early semantic parsing approaches require complex NLP pipeline components, such as POS tagger, template-fitting, relation-extraction. Berant et al. [10] train a semantic parser that scales up to Freebase with question-answer pairs instead of annotated logical forms. Reddy et al. [11] represent natural language via semantic graphs. They conceptualize semantic parsing as a graph matching problem. Fader et al. [12] propose an open question answering over curated and extracted knowledge bases. Berant et al. [13] also propose an approach that turns semantic parsing on its head via paraphrasing. With the maturity of deep learning, recent work has focused on building semantic parsing frameworks with neural networks to improve the effect. Xu et al. [14] propose SDP-LSTM to classify the relation of two entities in a sentence. Yih et al. [5] simplify the process of sematic parsing as the generation of the query graph. They train a CNN for the representation of the question and the relation. Dong and Lapata [15] propose an attention-enhanced encoder-decoder model to generate the logic form for the input utterance based on LSTM. Comparing with the end-to-end methods, semantic parsing can provide a deeper understanding of the question [4]. Moreover, process of answering which is staged can be easily for error analysis. In contrast, semantic parsing approaches usually have the disadvantage of error propagation and complicated steps.

Our approach is closely related to the second line of research with semantic parsing but simplifies the complicate steps as a semantic matching problem. Different from previous methods, we apply a type constraint to filter candidate answer entities.

3 Approach

We focus on the simple question that contains only one topic entity in this work. The problem is solved by the following two stages:

1. Mapping the words sequence in a question to a relation (or a relation chain with limited length, we use relation to refer both relation and relation chain for brevity) in the KB, it is also referred to *relation detection*. We reduce this problem to measuring semantic similarity of the words sequence and the relation.
2. Adding constraint. Inspired by [18], we find that answer type is also an important aspect for understanding question. Therefore, we take the entity type as a constraint to filter the correct answer entities from all candidates found by the first stage. We consider the process of *answer entity type prediction* as a binary classification problem.

For handling the multiple equivalent semantic statements of the same question, as well as the lexical gap between the natural language utterance and relation in the KB, we propose using Convolutional Neural Networks (CNN) [16,17,19] for relation detection and answer entity type prediction. The model is further elaborated in the next section (Sec. 3.1), followed by a description of the training process (Sec. 3.2).

3.1 Model Description

As illustrated in Fig. 1, under the precondition that the topic entity is identified by entity linking, our model consists of three components:

1. Two CNNs to produce low dimensional vector representations for the question (CNN_q) and the relation (CNN_r). For all relations starting from the topic entity in the

KB, we compare its vector representation with the representation of the question to detect the relation asked by the question. (Fig. 1a).

2. A CNN to predict answer entity types. Based on the topic entity and the detected relation, we form a query to search the KB for candidate answer entities. For all candidate types, we use CNN$_t$ to produce 2-dimensional vectors of question-type pairs for answer entity type prediction. (Fig. 1c).
3. The process of retrieving answers. We apply the predicted types as a constraint to filter correct answer entities from all candidates. (Fig. 1b).

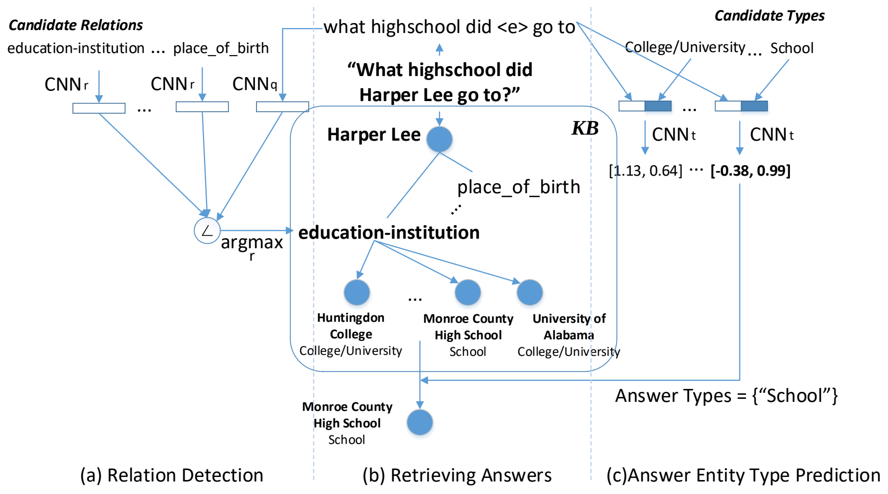
Candidate Relations
education-institution ... place_of_birth
CNNr
CNNr
CNNq
argmax r
what highschool did <e> go to
"What highschool did Harper Lee go to?"
KB
Harper Lee
place_of_birth
education-institution
Huntingdon College College/University
Monroe County High School School
University of Alabama College/University
Monroe County High School School
Candidate Types
College/University ... School
CNNt
CNNt
[1.13, 0.64] ··· [-0.38, 0.99]
Answer Types = {"School"}
(a) Relation Detection
(b) Retrieving Answers
(c)Answer Entity Type Prediction

Fig. 1. The overview of the proposed approach.

The details of each component are described in the following paragraphs.

Retrieving Answers. The process of retrieving answers (Fig. 1b) relies on relation detection and answer entity type prediction. Given a question q and the corresponding topic entity e_t, we present the question by replacing mention of e_t with symbol "<e>" for noise reduction. Then, our question answering approach proceeds in the following two stages:

1. Employ a relation CNN framework to detect the relation r for question q, and retrieve the candidate answer entities that has relation r connecting e_t in the KB. The candidate answer entities set is denoted as C_a^q.
2. Generate an answer entity type set T_a of q with the type prediction of another CNN framework, and remove the entities from C_a^q whose type is not in T_a to obtain the final answer entity set A_f.

Relation Detection. We construct an encoder-compare framework for relation detection. The process can be described as three steps as followed:

1. Generate the relation candidates set C_r^q for question q. We select all relations and chains of relations starting from the topic entity e_t. For a chain of relations, we limit its length to 2.
2. Extract k-dimentional semantic vectors for both q and each candidate relation in C_r^q by the relation CNN.
3. Calculate and then compare the similarity score between each candidate relation in C_r^q and q and take the top-scored one as the detected relation r.

The first step generates C_r^q including the relation chains which start from the topic entity. In Freebase's design, there is a special entity category called compound value type (CVT). Therefore, we consider the relation and relation chains (length $\leq$ 2) as the candidate. *The second step adapts a relation CNN framework which consists of two CNNs for extracting semantic vectors.* Fig. 2 shows the architecture of the CNN model for the question (CNN_q) and it is similar with the CNN_r for relations. Following the word hashing technique in [20], the model first applies a word hashing layer to construct the matrix for input word sequence. In this layer, a word is broken into a one-hot vector of letter-trigrams. For instance, the bag of letter-trigrams of the word "book" with the boundary symbol # consists of #bo, boo, ook, ok#. The word hashing layer is used due to two reasons. First, it can handle the problem of out of vocabulary (OOV) words. Second, letter-trigrams can grab some semantic information in English, such as "pre", "dis".

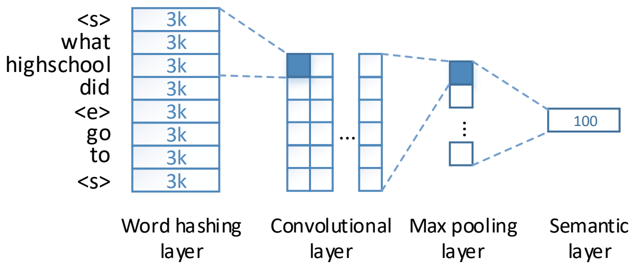

Fig. 2. The architecture of the CNN used for question to produce the vector.

After word hashing, the model use k kernels with a window size of three words to project the word sequence matrix to k vectors which hold the local features. Then, the max pooling layer extracts a global feature vector of length k by performing the maximum operation on each local feature vector. The following semantic (fully-connected) layer projects the global feature vector to a l-dimensional semantic vector of the question. We employ the CNN because its convolutional and max pooling layer can extract both local and global semantic features of word sequence, and make our model has more representation power.

*The third step detect*s the relation asked by the question. With the semantic vectors of both question q and candidate relation c_r, we compute the similarity score between p and c_r as follows:

$$S(q, c_r) = \cos(v_q, v_{c_r}) \quad (1)$$

where v_q and v_{c_r} are the semantic vectors of the q and c_r respectively, cos is the cosine similarity.

Answer Entity Type Prediciton. Although predicting answer entity types is still a matching problem of question and answer type, it different from relation detection. Because one question only corresponds to one relation but can correspond to more than one answer type. For instance, for question "where is the fukushima daiichi nuclear plant located?" the two answer entities Japan and Okuma have different types "Country" and "City/Town/Village". Therefore, we consider the matching problem as a task of binary classification for each question-type pair, which is proceeded as the following three steps.

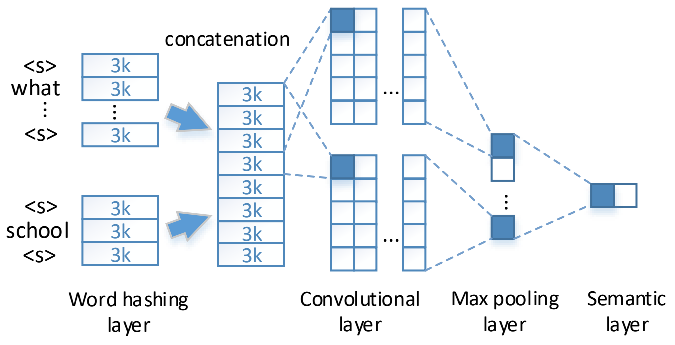

Fig. 3. The architecture of the type CNN.

The first step generates a type candidates C_t^q consists of type labels for the question q. We collect the lowercased English type labels (for Freebase, we use the **common.topic.notable_types** property) for each candidate answer entity generated by the topic entity and the detected relation from the KB, and take them as candidate types.

The second step concatenates q and each type in C_t^q to a new word sequence and feed it into the type CNN to output a prediction vector indicates whether the candidate type is positive. The type CNN framework is illustrated in Fig. 3. Different from the CNN for relation detection, we concatenate the sequence matrix of question and type and feed it to the type CNN. Moreover, the type CNN uses the kernels with two kinds of window sizes, three words and four words respectively to extract local features. The output of CNN is a 2-dimensional vector $v = [v_0, v_1]$ which is the probability distributions of the matching. The candidate type of the question is judged as positive when v_1 is greater than v_0.

The third step generates an answer entity type set T_a which consists of all positive types in C_t^q.

3.2 Training

We treat relation detection as a multi-class classification problem while answer entity types prediction as a binary classification problem. So we train the relation CNN and the type CNN in different ways. We detail the loss functions and negative sampling strategies in the next few paragraphs.

Loss Function. For the matching of question and relation, we drive the model to output a higher similarity score for the correct pair than wrong pairs. We consider each candidate relation as a class and use the cross entropy loss function minimized during training is given by

$$\sum_{q \in Q} < y'_q, \log(y_q) > \tag{2}$$

That is, Q is the set of all questions in the training set and $< x, y >$ is a dot product operation of two vectors. y'_q is a one-hot vector that y'_{q_i} is 1 when the i-th candidate relations c_r^i is the correct relation. y_q is a probability distributions of candidate relations, that is, y_{q_i} is the probability that c_r^i is the relation asked by the question q, which is calculated as

$$y_{q_i} = \frac{\exp(\cos(v_q, v_{c_r^i}))}{\sum \exp(\cos(v_q, v_{c_r^j}))} \tag{3}$$

where v_q and $v_{c_r^i}$ are semantic vectors of the question q and the i-th candidate relations c_r^i respectively.

For predicting the answer types, we project the concatenated word sequence for each question-type pair to a 2-dimensional vector v by the type CNN. Then, we normalize v to get a probability distributions y_{pt} by the softmax function. The loss function for training pattern-type pairs is given by

$$\sum_{q \in Q, t \in C_t^q} < y'_{q,t}, \log(y_{q,t}) > \tag{4}$$

where $y'_{q,t}$ is a 2-dimensional vector that indicates whether t is a positive type or not.

Negative Sampling. For training CNN, the positive relation sample s_r^+, which is the correct relation r of a question q, can be obtained from the sematic parses provided in the training data. We can also collect the correct answer entity types t_a as positive type samples set T_a^+ of a question q from the training set. However, the dataset does not provide the negative samples for training. Therefore, we need perform negative sampling from the KB. The generation of corrupted samples are based on s_r^+ and T_a^+. We observe that the way of negative sampling has a significant influence on results. In principle, we try to simulate a condition of training which is close to the condition during the test.

Relation Corruption. To corrupt s_r^+, a false relation can be sampled from the space of all relations in the KB. However, we tend to make the model more distinguishable from more plausible wrong relations. We use the same candidate relation generation procedure as the one used for test. The negative relation sample set N_s is generated as

$$N_s = C_r^q - \{s_r^+\} \tag{5}$$

where C_r^q is the candidate relation set for a question q in the training set.

Type Corruption. We collect the type label of Freebase entity by its **common.topic.notable_types** relation. To make the model more distinguishable, we train the type CNN to predict correct answer entity types from the candidate types C_t^q. The negative type sample set N_t is generated as

$$N_t = C_t^q - T_a^+ \tag{6}$$

where C_t^q is a type set of all the entities in C_a^q. These entities are retrieved along the relation s_p^+ from the topic entity in the KB.

In our experiments, we observed that the number of negative type samples is about one third of the number of the positive type samples. The unbalance in quantity between positive and negative samples leads to the result that the model tends to judge negative sample as a positive one. So, we adapt a simple over-sampling strategy that training three times on N_t while one time on T_a^+ at each iteration. Then, we achieve a significant better result on answer entity type prediction than before.

4 Experiments

In this section, we detail the experiments we made to evaluate our model. First we introduce the dataset and evaluation metric, followed by the main experimental results and error analysis.

4.1 Data and Evaluation Metric

We evaluate the proposed approach on the WEBQUESTIONSSP dataset [6], which contains full semantic parses for a subset of the questions from WEBQUESTIONS dataset [10]. It removes 18.5% of the origin dataset because these questions are considered as "not answerable". The dataset consists of 4737 question-answer pairs which are split into training set of 3098 and test set of 1639. The questions of dataset were collected using Google Suggest API and the answers were obtained using Amazon Mechanical Turk workers. They were updated in [6] using Freebase and added full semantic parses.

In our experiments, we directly obtain the topic entity of each question by the entity linking results from the semantic parses provided in the dataset for both training and testing. In addition, we drop some questions which have no answers or no relation during training, and out model outputs empty answer entity sets for this kind of questions on the evaluation stage. The system performance is basically measured by the ratio of questions that are answered correctly. Because there can be more than one answer to a question, *precision*, *recall*, and F_1 metrics are computed based on the output of each individual question, and average F_1 is reported as the main evaluation metric. We compute experiment results with the official evaluation script[1].

4.2 Experimental Settings

We add boundary symbols "<s>" at the beginning and end of all word sequences. In order to solve the difference in word sequence length, we specify that each question has a length of 20 words, each candidate relation has a length of 4 words, each question-type pair has a length of 25 words. For word sequences of insufficient length, we add all-zero letter-trigrams vectors at the end of the word hashing matrix to reach the specified length.

Because the CNNs of our model are used for different tasks, we do not set the hyper-parameters for training them in the same way. For the relation CNN, the convolutional layer size is set to 10000, the max pooling layer size is set to 500, the semantic (output) layer size is set to 100, the learning rate is 0.01 and the training process lasts 200 epochs. For the type CNN, the convolutional layer size is set to 20000, the max pooling layer size is set to 2000, the semantic (output) layer size is set to 2, the learning rate is 0.001 and the training process lasts 200 epochs.

Table 1. Comparison of our approach with existing work.

Method	Prec.%	Rec.%	Avg. F_1%
Yih et al. 2016 (Answers)[6]	67.3	73.1	66.8
Liang et al. 2017[21]	70.8	76.0	69.0
Yih et al. 2016 (Sem. Parses)[6]	70.9	80.3	71.7
Our approach (Rel. Detection)	68.9	82.4	70.8
Our approach (Rel. Detection+Type Prediction)	79.9	82.3	**74.8**

4.3 Results

Table 1 shows the overall results of our approach compared to existing work [6,21] on the WEBQUESTIONSSP dataset. The results show that our approach outperforms the previous state-of-the-art method by 3.1% in average of F_1 measure.

The main difference of our approach comparing with previous methods is that we use the type constraint on candidate answer entities to improve the precision. Therefore, for examining the contribution of the type constraint, we test the performances with and without the type constraint in Table 1. When the retrieval of answers relies only on the relation detection, the performance of our system is close but does not exceed previous work. After adding the predicted entity type to constrain candidate answers, the

1 Available at http://aka.ms/WebQSP

performance of the whole approach increases 4% in F_1. The 10% increase in precision indicates that our type constraint is effective on removing wrong answers.

For the perspective of the neural networks, the relation CNN achieves 79.7% accuracy on relation dectection, and the type CNN achieves 91.1% in F1 (Prec. = 90.5%, Rec. = 94.1%) on predicting answer entity types for all questions in the test data. For examining the contribution of the word hashing layer in CNN, we test the performances of the relation CNN on word-level and character-level. On word-level, we use *GloVe* [22] vectors provided on the Glove website[2] as word embeddings. On character-level, we first consider both question and relation as character sequences. Then, we use the character vectors in Glove to construct the matrix for input character sequence. The results are shown in Table 2. Both word and character-level are far less effective than word hashing technique on the proposed relation CNN.

Table 2. The performances of the relation CNN on word-level and character-level.

Granularity	Acc. %
Word-level	52.3
Character-level	55.8
Word-hashing	79.7

4.4 Error Analysis

Although our approach substantially outperforms existing methods, it is partly because we directly use the results of entity linking provided in the dataset and there is no error on obtaining topic entity.

For error analysis, we randomly sampled 100 questions from 500 questions which our system did not output the completely correct answer set. We find that 57% of the errors are due to the incorrect relation detection, 23% are due to ignoring the time constraints of some questions. For example, for question “What team does Jeremy Lin play for *2013*?” we output all teams that Jeremy Lin played for without considering the time constraint. We don’t consider other similar time constraints either, such as “first”, “now”. Existing more than one entity in a question like “who plays *Lex Luthor* on *Smallville*?” also causes 8% of the errors. 3% of the errors are due to the types used for constraint are too wide to remove wrong answers. For instance, the answer entity type we predicted for question “What is the state flower of New Mexico?” is “Official Symbol”. The rest 9% errors are caused that the answers retrieved from the KB are not completely same with the gold answers when all stages are correct.

5 Conclusion

In this paper, we present a staged, convolutional neural network-based approach with type constraint for question answering over knowledge base. In the first stage, we detect

[2] https://nlp.stanford.edu/projects/glove/

the relation that starts from the topic entity to obtain candidate answer entities. We also add the type constraint on candidate answer entities to remove wrong answer entities in the second stage. With the help of the convolutional neural network model and topic entity linking results provided in the dataset, our approach outperforms the previous method on the WEBQUESTIONSSP dataset.

Acknowledgements

The work is supported by the Natural Science Foundation of China under grant No. 61502095, and the Natural Science Foundation of Jiangsu Province under Grant BK20140643.

References

1. K. Bollacker, C. Evans, P. Paritosh, et al.: Freebase: A collaboratively created graph database for structuring human knowledge. In: proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp 1247–1250. ACM, New York, USA (2008).
2. S. Auer, C. Bizer, G. Kobilarov, et al.: DBpedia: A nucleus for a web of open data. The Semantic Web, Springer, 722–735 (2007).
3. Y. Xiao, H. Wang, Y. Song, S. Hwang and W. Wang. KBQA: Learning Question Answering over QA Corpora and Knowledge Bases. In: Proceedings of the 42nd International Conference on Very Large Data Bases, pp. 565–575. ACM, New Delhi, India (2016).
4. L. Denis, F. Asja, et al.: Neural Network-based Question Answering over Knowledge Graphs on Word and Character Level. In: 26th International World Wide Web Conference, pp. 1211–1220. Perth, Australia (2017).
5. S.W.t. Yih, M.W. Chang, et al.: Semantic parsing via staged query graph generation: Question answering with knowledge base. In: Proceedings of the 53nd Annual Meeting of the Association for Computational Linguistics, pp. 1321–1331. Association for Computational Linguistics, Beijing, China (2015).
6. W. Yih, M. Richardson, C. Meek, et al.: The Value of Semantic Parse Labeling for Knowledge Base Question Answering. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pp. 201–206. Association for Computational Linguistics, Berlin, Germany (2016).
7. Z. Dai, L. Li, W. Xu: CFO: Conditional focused neural question answering with large scale knowledge bases. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pp. 800–810. Association for Computational Linguistics, Berlin, Germany (2016).
8. Mo Yu.: Improved neural relation detection for knowledge base question answering. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pp. 571–581. Association for Computational Linguistics, Vancouver, BC, Canada (2017).
9. Y. Qu, J, Liu, L. K, et al.: Question Answering over Freebase via Attentive RNN with Similarity Matrix based CNN. arXiv preprint arXiv: 1804.03317 (2018).
10. J. Berant, A. Chou, R. Frostig, and P. Liang.: Semantic parsing on freebase from question-answer pairs. In: Proceedings of the 2013 Conference on Empirical Methods in Natural

Language Processing, pp. 1533–1544. Association for Computational Linguistics, Seattle, Washington, USA (2013).

11. S. Reddy, M. Lapata, and M. Steedman.: Large-scale semantic parsing without question-answer pairs. Transactions of the Association for Computational Linguistics, 2(2014) 377–394 (2014).
12. A. Fader, L. Zettlemoyer, and O. Etzioni.: Open Question Answering over Curated and Extracted Knowledge Bases. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 1156–1165. ACM, New York, USA (2014).
13. J. Berant and P. Liang.: Semantic parsing via paraphrasing. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pp. 1415–1425. Association for Computational Linguistics, Baltimore, Maryland (2014).
14. X. Yan, L. M, G. Li, et al.: Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Path. In: The 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1785–1794. Association for Computational Linguistics, Lisbon, Portugal (2015).
15. L. Dong and M. Lapata.: Language to Logical Form with Neural Attention. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pp. 33–43. Association for Computational Linguistics, Berlin, Germany (2016).
16. D. Zeng, K. Liu, S. Lai et al.: Relation Classification via Convolutional Deep Neural Network. In: The 25th International Conference on Computational Linguistics: Technical Papers, pp. 2335–2344. ACM, Dublin, Ireland (2014).
17. Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil.: A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pp. 101–110. ACM, Shanghai, China (2014).
18. L. Dong, F. Wei, M. Zhou and K. Xu.: Question Answering over Freebase with Multi-Column Convolutional Neural Networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pp 260–269, Beijing, China, (2015).
19. J. Gao, P. Pantel, M. Gamon, X. He, L. Deng, and Y. Shen. Modeling Interestingness with Deep Neural Networks. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (2014).
20. P. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck.: Learning Deep Structured Semantic models for Web Search using Clickthrough Data. In Proceedings of the 22nd ACM International Conference on Conference on information and knowledge management, pp. 2333-2338. ACM, San Francisco, CA, USA (2013).
21. C. Liang, J. Beranty, Q. Le, K. D. Forbus, and N. Lao. Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pp. 23–33. Association for Computational Linguistics, Vancouver, BC, Canada (2017).
22. J. Pennington, R. Socher, and C. D. Manning. Glove: Global Vectors for Word Representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pp 1532–1543. Association for Computational Linguistics, Doha, Qatar (2014)