L2R-QA: An Open-Domain Question Answering Framework

Tieke He[1], Yu Li[1], Zhipeng Zou[1], and Qing Wu[2](✉)

[1] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, People's Republic of China
hetieke@gmail.com
[2] School of Economics, Nanjing 210093, People's Republic of China
wuqing@nju.edu.cn

Abstract. Open-domain question answering has always being a challenging task. It involves information retrieval, natural language processing, machine learning, and so on. In this work, we try to explore some comparable methods in improving the precision of open-domain question answering. In detail, we bring in the topic model in the phase of document retrieval, in the hope of exploiting more hidden semantic information of a document. Also, we incorporate the learning to rank model into the LSTM to train more available features for the ranking of candidate paragraphs. Specifically, we combine the results from both LSTM and learning to rank model, which lead to a more precise understanding of questions, as well as the paragraphs. We conduct an extensive set of experiments to evaluate the efficacy of our proposed framework, which proves to be superior.

Keywords: Question answering · Learning to rank · Topic model

1 Introduction

Since the 1960's, a large number of Question answering systems has been developed [1]. It addresses different types of questions on different domains from different data sources. We focus on the answer of factoid open-domain questions adopting Wikipedia as the knowledge source. Open-domain means the domain of questions is not limited, and its development was limited because of the absence of a viable knowledgebase. The emergence of Wikipedia, a simultaneously evolving collection of information on diverse topics, provides an opportunity to bridge the gap between the open-domain questions and knowledge base. According to Jurczyk's analysis [9], Wikipedia performs well on different types of questions. It is a suitable knowledge base for what, how and who questions. Especially for what- questions, where the coverage of answers can reach 60%. And for the other two types of questions, Wikipedia also performs better than other existing knowledge bases. Besides, Wikipedia has diverse types of data which maintains high coverage on numerical, personal, and objective facts.

Using the large-scale collection of documents brings the challenge of open-domain question along with machine reading at scale (MRS). Traditional reading comprehension supposes that the input text contains the answer, while it is not realistic for open domain QA. We need to narrow down the range of answers, i.e., selecting the most relevant documents first and then using reading comprehension to process. Besides, limiting to the single knowledge source force the model to be very precise because the answer may only appear once.

To alleviate all these problems, we propose a new open-domain question answering framework based on Wikipedia text. We mainly follow the structure as Zhao et al. [4] proposed to solve the problem of machine reading at scale. However, in their framework, they use the TF-IDF for the representation of documents, which may lose much semantic information of the documents, leading to inaccuracy when computing the similarities between documents. In seeing this, we try to enhance that by adopting some advanced models that better represent these documents, i.e., Latent Semantic Indexing (LSI) for example, and then we calculate the similarity between them to get the top 5 relevant documents according to given questions. Also, we consider more features during the reading comprehension phase, especially, our framework adds up the learning to rank (LTR) as features to better encode both the paragraph and question, in order to find the final answer more precisely. An extensive set of experiments are designed and conducted to testify the efficacy of our proposed framework, and the results demonstrate the superiority of it.

The main contributions of our work are as follows:

– We apply LDA on document representation in the task of large-scale document retrieval and compare with the accuracy and efficiency while using the tfidf model.
– We use the attention mechanism in selecting answers from candidates by using learning to rank (LTR) model to pay more attention to answers included in more relevant documents.

The rest of the paper is organized as follows: In Sect. 2, we introduce the related work. Section 3 presents our framework for open-domain question answering, the methods and techniques we used are depicted in details in this section. Section 4 describes four datasets used in our work. The details of our experiments and the evaluation part are in Sect. 5. Comparison with others' work is also presented in this section. We conclude our work In Sect. 6, along with future work.

2 Related Work

2.1 Question Answering Systems (QASs)

Question answering systems (QASs) was born to solve the "Last-Mile problem" of the search engines. Instead of presenting an ordered list of related documents or corresponding web links, the QASs generate the answers of the question asked in natural language directly, which save the time of browsing the documents to find the answer.

Based on the type of data source, QASs can be classified as knowledge-base question answering (KB-QA) and text-based question answering. For KB-QA, whether using traditional methods, such as semantic parsing [2], information extraction [15], vector modeling [3], or using deep learning methods to improve it, such as adapting Convolutional Neural Network (CNN) on the semantic analysis method [16] and vector modeling method [7], using Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) to classify entity relationship [14] can not solve the fixed schemes and incompleteness. The incompleteness makes the system unable to use these knowledge bases. Hence the corresponding answer cannot be given. Besides, the reason for incompleteness including real lacking raw data and not mined data cannot be determined, which adds difficulties for optimization and improvement. On the other hand, triples used to represent questions cannot faithfully represent the semantic structure of the question, especially for more complicated question or its concept is rather vague. However, answering from the raw text as text-based QA avoid these problems, which motivate the proposal of our method.

2.2 QA with Wikipedia

Wikipedia is a collaborative, continuously growing, semi-structured knowledge source which is usually applied to QA as the knowledge base. Jennifer leverages the characteristics of Wikipedia to implement type independent candidate generation method for QA, which shows that Wikipedia metadata can be used to extract candidate answers from results returned by searching without inputting ontology. Pum-Mo Ryu makes full use of Wikipedia by using different types of data to answer different kinds of question [5]. They suggest that infoboxes, category structure, and definitions have their unique strength in answering factoid questions, list questions and descriptive questions perspectively. And the combination of answers from different modules using different types of semi-structured knowledge source has been improved compared with the traditional use of Wikipedia [6,13]. Wikipedia is also used as an auxiliary data source. Sergio presents a novel architecture to solve the Cross-Lingual Question Answering task with the use of multilingual relations encoded in Wikipedia when processing Name-Entities in queries [8].

In our work, we treat Wikipedia as a collection of documents and use LDA and LSI to filter the relevant documents of question from more than 5 million items. And the result is the input of the machine comprehension of text to find the final answer.

3 Datasets

3.1 Wikipedia

We use the processed Wikipedia provided by Danqi et al. (2017) as our knowledge base to find answers in the full-scale experiment. The document dump includes the latest documents of diverse topics [13]. Based on the 2016-12-21 dump of English Wikipedia, they used the WikiExtractor script to extract and clean text for machine comprehension. The output document files of the extractor are consist of a serious of articles which are represented by XML element. And each element has two attributes: (1) id, which is the identification of the article (2) URL, which is the link to the original Wikipedia page, including content that contains pure text, one sentence per line. Note that the first sentence is the title of the article.

3.2 SQuAD

The Stanford Question Answering Dataset (SQuAD) dataset is a large-scale reading comprehension dataset generated by crowdsourcing. Rajpurkar et al. [12] used Project Nayuki's Wikipedia internal PageRank to get top 10000 articles from Wikipedia and sampled 536 articles randomly as the resource, and then filtered to get 23,215 paragraphs which cover a wide range of topics. For each paragraph, up to 5 question and answering tasked to be given based on the content. Furthermore, the crowd workers are tasked to select the shortest span based on the question along with the paragraph. Hence, each example in SQuAD consist of articles and each article consists of title and paragraphs along with human-generated questions and corresponding answers in the type of span usually. Besides, they used two metrics in evaluation:

1. Exact match (EM), which is used to measure the percentage of matching where the prediction matches one of the ground truth exactly.
2. (Macro-average) F1 score, which is used to measure the average overlap between them.

In our work, we use SQuAD which contains 87k examples in training and evaluating periods of selecting an answer from candidate paragraphs returned by document retrieval to complete the reading comprehension task. And the development SQuAD containing 10k examples is used only in evaluating period of open-domain question answering, which means giving the whole Wikipedia as the resource to find the answer.

4 Framework

In our framework, we use information retrieval, topic model, bidirectional long short-term memory network (BiLSTM) and feature engineering et al. to implement answering questions based on large-scale documents. Figure 1 gives an illustration of our system.

4.1 Information Retrieval and Topic Model

In our work, we use LSI and LDA to replace TF-IDF used in Chen's DrQA [4]. Which improves efficiency and accuracy.

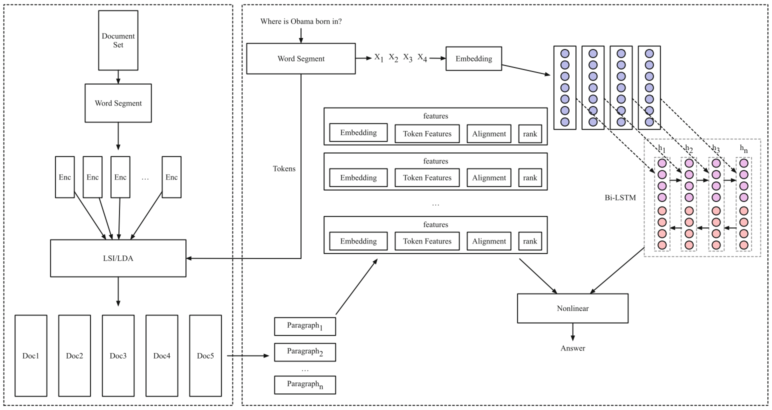

Fig. 1. The proposed framework L2R-QA

In the module of document retrieval, we use LDA to model the documents extracted from the Wikipedia and the given question. The integrity process flow is as follows.

- Use CoreNLP toolkit [10] to tokenize, name entity tags, generate lemma and part-of-speech.
- Construct a bag-of-words model and index of the document.
- Compute TF-IDF of the document.
- Model the documents with LSI.

When retrieving relevant documents of given question, we first model the question with LSI model generated, and then return the top 5 most relevant documents based on the similarity between question and documents.

4.2 Paragraph Features Extraction

In our work, we use paragraph as the smallest units of data for machine comprehension, and apply BiLSTM to process the features of paragraph to learn the model. We describe the details of our methods as follows.

We divide the documents returned by document retriever into paragraphs and denotes it as Eq. 1

$$A = \{p_1, p_2, \cdots, p_m\} \tag{1}$$

And tokenize the paragraphs as Eq. 2

$$p_i = \{t_1, t_2, \cdots, t_n\} \tag{2}$$

For each token t_i, we use five methods to extract features, including four features has been used in DrQA and additional one added by us. Hence, we introduce the first four features briefly and the added one in details.

- Word embedding: We used the fine-tuned word embedding based on the 300-dimentional Glove word embedding, which consider the keywords like what, when, who, that are crucial in QA. And denote it as Eq. 3

$$v_{embedding}(p_i) = E(p_i) \tag{3}$$

- Exact match: 3 binary features are used to indicate whether p_i can match any word in question q in original, lower-case or lemma form respectively. And denote it as Eq. 4

$$v_{exact-match} = M(p_i \in q_i) \tag{4}$$

- Token features: Include three manual factors include part of speech (POS), name entity recognition (NER) and normalized term frequency (TF), which is denoted as Eq. 5

$$v_{token-feature}(p_i) = (POS(p_i), NER(p_i), TF(p_i)) \tag{5}$$

- Aligned question embedding: Consider the doft alignment between the words with similar but not same word, Chen adds aligned question embedding. Denote as Eq. 6

$$v_{align}(p_i) = \sum_j a_{i,j} E(q_j) \tag{6}$$

where

$$a_{i,j} = \frac{exp(\alpha(E(p_i)) \cdot \alpha(E(q_j)))}{\sum_{j'} exp(\alpha(E(p_i)) \cdot \alpha(E(q_{j'})))} \tag{7}$$

Thus, we extracted all the features to represent the topic, semantic and the semantic structure of the paragraph, which is denoted as P_i. Hence, the representation of article can be denoted as $\boldsymbol{A} = \sum_1^m \mathbf{p}_i$. We also choose BiLSTM to learn the presentation of paragraph, and update the value of hidden weights step by step.

$$\boldsymbol{q} = \mathbf{BiLSTM}(\{\boldsymbol{q_1}, \boldsymbol{q_2}, \cdots, \boldsymbol{q_l}\}) \tag{8}$$

4.3 Question Semantic Modeling

Question is usually a short sentence which has simple semantic structure and few implicit semantic. Recurrent Neural Network can perform well on modeling the semantic of the question, and BiLSTM can improve the performance. In last section of our system, we tokenize question into sequences and denote it as $q = \{q_1, q_2, \cdots, q_k\}$. We suppose that $\boldsymbol{q} = nonlinear(\{\boldsymbol{q_1}, \boldsymbol{q_2}, \cdots, \boldsymbol{q_k}\})$,where $\boldsymbol{q}$ is the semantic vector of question, and $\boldsymbol{q_i}$ denotes the vector of *ith* word in question. We apply another BiLSTM on question embedding as Eq. 9:

$$\boldsymbol{q} = \mathbf{BiLSTM}(\{\boldsymbol{q_1}, \boldsymbol{q_2}, \cdots, \boldsymbol{q_l}\}) \tag{9}$$

Recurrent neural network (RNN) can contact the context to express the semantics of text and can perform well in modeling text. While LSTM pays more attention on the context related to the current word, i.e. the importance of words differs in different sentences, which is also known as keyword effect. So we need to set different weights on different words to accurately represent the semantic features of the text as $\boldsymbol{q} = \sum_k w_k \boldsymbol{q_k}$. For w_k, we can get Eq. 9, where u is the shared parameters learned by BiLSTM.

$$w_k = nonlinear(u \cdot q_k) \tag{10}$$

4.4 Candidate Answers Selection

Candidate answer selection is to find the final answer from the paragraph retrieved by the system in the first period. The framework of selecting answer is shown in Fig. 1. Generally speaking, we combine the features of questions and paragraph tokens and use deep learning algorithms to learn the weights. After feature extracting and parameters learning of paragraphs, sequences and questions, the system can represent the semantic of them very well. We use the feature vector of tokens and questions as input and train another two BiLSTMs to get the start token and end token of answer independently. Then the content between the start and end token is supposed to be the final answer. To capture the similarity between question $\boldsymbol{q}$ and paragraph p when each token being start and end, we compute the probabilities using nonlinear terms as

$$P_s(m) = nonlinear(t_m W_s \boldsymbol{q}) \tag{11}$$

$$P_e(n) = nonlinear(t_n W_e \boldsymbol{q}) \tag{12}$$

The original method is to maximize the $P_s(m) \times P_e(n)$ t get the start and end tokens, while we consider the effect of the paragraph using the results of learning to rank (LTR). The answer is supposed to be contained in the paragraph, so take paragraph as the smallest unit, the paragraph which has a higher score in LTR is more likely to contain the correct answer. In the candidate answer selection, we use LTR as part of weight when computing the probabilities of start and end tokens. In other words, the tokens contained in the top paragraph in LTR are rewarded to increase the possibility of choosing the final answer. And the reward is a_i, and the probability to be maximized changed to Eq. 13

$$P_{i,m,n} = \max(a_i P_s(m) P_e(n)) \tag{13}$$

In summary, we use information retrieval and current neural network to complete the machine learning at scale task. Firstly, we tokenize the documents of Wikipedia and question to get the valid sequences. Then use the BoW method [17] to encode the terms and TF-IDF to calculate the importance of terms. With the statistics of TF-IDF, we apply LDA/LSI to model the document features, i.e, using Dirichlet Probability Model/Singular Value Decomposition to compute the similarity between documents and given questions. The top 5 relevant documents are returned as the input of reading comprehension.

In the machine reading period, we divide the document into paragraphs for processing. With the same method used to tokenize in document retrieval period, and construct the features of terms manually. For each term in the paragraph, it has the feature of word embedding vector. At the same time, we add NER and POS tagging process to get the features. These features can specifically show the type of words and the location of words, which can provide a basis for the neural network to understand the semantic structure. Besides, we add LTR to improve the importance and attention of paragraphs as shown in Fig. 2. For the generated feature vectors, we use BiLSTM network to learn the rules corresponding to the segmentation vectors as the final feature of the text. As for the question, we apply another BiLSTM on tokenized text to get the semantic vector $\boldsymbol{q}$.

In answer selection period, we use two independent neural networks to find the starting and ending positions of the answer and use a nonlinear model to calculate the probability that each word in the candidate answer becomes the beginning and the end. Additionally, we also considered the influence of LTR's paragraph ranking on the answer.

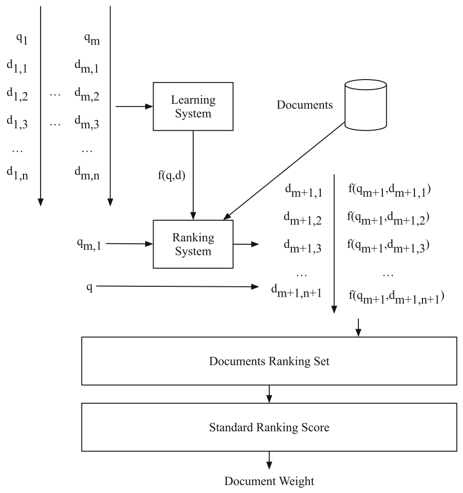

Fig. 2. Learning to Rank in our work

5 Experiments

We introduce our experiments from three aspects: (1) document retrieval based on LSI and LDA (2) reading comprehension of paragraphs and (3) open-domain question answering system with the combination of (1) and (2).

5.1 Document Retrieval Based on LSI

First, we use gensim [11] to learn a topic distribution model from the SQLite-formatted document library[1]. Based on this model, we evaluated our document

[1] docs.db provided by Wikipedia's open source Document Dump here.

retrieval module on all the datasets which contain SQuAD and expanded CuratedTREC, WebQuestions, WikiMovies. Table 2 shows the examples of wiki documents found by our retriever according to the given question. And Table 4 shows the latent semantic found by our Lsi model from the aspect of weight in question and weight in Wikipedia documents respectively, while also taking the question in Table 2. As for the evaluation, we use the hit probability of top 5 document, i.e. the ratio of top 5 documents that contains the correct answer.

Table 1. Documents retrieval results

Question	*Wiki Documents*
Who is the president of the United States?	List of Presidents of New York University
	Vice President of Madagascar
	Vice President of Madagascar
	List of United States Presidential firsts
	...others

Table 2 shows the latent semantic found by our LSI model from the aspect of weight in question and weight in Wikipedia documents respectively, while also taking the question in Table 1 as an example. As shown in table, there are not only words in question, but some synonyms.

Table 2. Latent semantic found by LSI model.

Score type	*Word*	Score
Weight in question	President	0.949
	America	0.033
Weight in Wikipedia documents	President	0.949
	President	+0.959
	Prime	+0.003
	Presidential	+0.001
	Chairman	+0.001

5.2 Document Reader

Our reading model was trained and evaluated on SQuAD data set. We use the Stanford CoreNLP toolkit to tokenize, name entity recognize and encode the paragraphs and given questions. The obtained word vectors are sent to a three-layer BiLSTM neural network model. In order to avoid overfitting, we set the dropout rate of 0.3, i.e. 30% neural network units are randomly discarded in each batch of training.

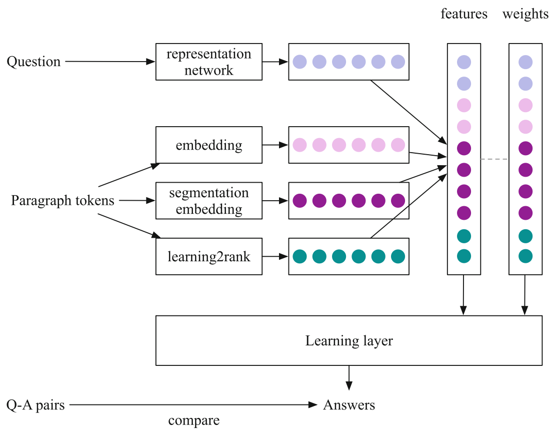

Fig. 3. The **picker** of candidate answers

We evaluated our module of machine comprehension on the SQuAD test set. We compare the results with the results of the Document Reader module in DrQA on the SQUAD test set. As can be seen from the comparison data, our module performed slightly better, indicating that the CTR feature we added during the training period had a positive effect. Although the effect is not very significant, it is not bad. The framework of selecting answer is shown in Fig. 3. Generally speaking, we combine the features of questions and paragraph tokens and use deep learning algorithms to learn the weights. They are encoded as vectors, and we merge all vectors. Then there is a weight vector to learn in the learning layer, which will be updated in the layer.

5.3 Open-Domain Question Answering System

Open-domain question answering system combines the tasks of document retrieval and reading comprehension. According to the input question, the document retrieval part will select five documents most relevant to the problem in the collection of Wikipedia documents and return it to the reading part for reading comprehension. As for reading comprehension period, documents are first divided into paragraphs. Then we choose the most relevant paragraph and locate the answer in it. Both the paragraph and the predictable answer will be returned as a result. We evaluate our system on all four datasets with Wikipedia as the single knowledge source.

According to the results of Chen's experiments, we can see the EM indicator has dropped significantly, from **69.5** to **70.0** respectively, which also appears in our experiments. In the experiment of machine comprehension, the data we give is a paragraph that is clearly related to the problem, that is, the answer is contained in the paragraph. While for the whole system, we give the entire collection of Wikipedia documents. The retriever module returns five documents that are most relevant to the question, but the correct answer may not be in these five documents, which leads to a decline in performance. That is the reason why we are interested in the improvement of retrieval module and make changes on it. Also, there is still room for improvement in this part, and we still need to work harder to move further in improving the relevance of the documents returned by the retrieval module to the problem.

We also compare our system with DrQA on four datasets, which indicates that both using LSI in the retrieving period and add learning to rank in reading comprehension period has a positive effect. Later, we may consider a more granular granularity. For example, the retrieval model returns paragraphs rather than documents, so that the reading model analyzes tasks based on paragraphs and gives predictions, or apply Learning to Rank method on sentences rather than paragraphs. We will continue to work in this area, and hope to get better results.

6 Conclusion

In this work, we propose an enhanced framework for the open-domain question answering task, in which, we introduce some topic model during the document retrieval phase, leading to a better representation of the documents, i.e., more hidden semantic information is exploited. Moreover, we incorporate the learning to rank model into the LSTM to train more available features for the ranking of candidate paragraphs. Specifically, we combine the results from both LSTM and learning to rank model, which lead to a more precise understanding of questions, as well as the paragraphs. The result of the empirical experiment demonstrates the efficacy of our proposed framework. We would like to test the framework with more advanced topic models and ranking methods in the future, as well as more knowledge bases.

Acknowledgement. The work is supported in part by the National Key Research and Development Program of China (2016YFC0800805) and the National Natural Science Foundation of China (61772014).

References

1. Androutsopoulos, I., Ritchie, G.D., Thanisch, P.: Natural language interfaces to databases-an introduction. Nat. Lang. Eng. **1**(1), 29–81 (1995)
2. Berant, J., Chou, A., Frostig, R., Liang, P.: Semantic parsing on freebase from question-answer pairs. In: Proceedings of EMNLP (2013)
3. Bordes, A., Chopra, S., Weston, J.: Question answering with subgraph embeddings. Comput. Sci. (2014)
4. Chen, D., Fisch, A., Weston, J., Bordes, A.: Reading Wikipedia to answer open-domain questions, pp. 1870–1879 (2017)
5. Chu-Carroll, J., Fan, J.: Leveraging Wikipedia characteristics for search and candidate generation in question answering. In: AAAI Conference on Artificial Intelligence, pp. 872–877 (2011)

6. Denoyer, L., Gallinari, P.: The Wikipedia XML corpus. In: Fuhr, N., Lalmas, M., Trotman, A. (eds.) INEX 2006. LNCS, vol. 4518, pp. 12–19. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73888-6_2
7. Dong, L., Wei, F., Zhou, M., Xu, K.: Question answering over freebase with multi-column convolutional neural networks. In: Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing, pp. 260–269 (2015)
8. Ferrández, S., Toral, A., Ferrández, Ó., Ferrández, A., Munoz, R.: Exploiting Wikipedia and EuroWordNet to solve cross-lingual question answering. Inf. Sci. **179**(20), 3473–3488 (2009)
9. Jurczyk, T., Deshmane, A., Choi, J.D.: Analysis of Wikipedia-based corpora for question answering (2018)
10. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., Mcclosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Meeting of the Association for Computational Linguistics: System Demonstrations (2014)
11. Řuřek, R., Sojka, P.: Gensim–statistical semantics in python (2011)
12. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100,000+ questions for machine comprehension of text, pp. 2383–2392 (2016)
13. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014)
14. Yan, X., Mou, L., Li, G., Chen, Y., Peng, H., Jin, Z.: Classifying relations via long short term memory networks along shortest dependency path. Comput. Sci. **42**(1), 56–61 (2015)
15. Yao, X., Durme, B.V.: Information extraction over structured data: question answering with freebase. In: Meeting of the Association for Computational Linguistics, pp. 956–966 (2014)
16. Yih, W.T., Chang, M.W., He, X., Gao, J.: Semantic parsing via staged query graph generation: question answering with knowledge base. In: Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing, pp. 1321–1331 (2015)
17. Zhao, R., Mao, K.: Fuzzy bag-of-words model for document representation. IEEE Trans. Fuzzy Syst. **26**, 794–804 (2017)