Question Answering with Character-Level LSTM Encoders and Model-Based Data Augmentation

Run-Ze Wang, Chen-Di Zhan, and Zhen-Hua Ling

National Engineering Laboratory for Speech and Language Information Processing,
University of Science and Technology of China, Hefei, China
{wrz94520,cdzhan}@mail.ustc.edu.cn,zhling@ustc.edu.cn

Abstract. This paper presents a character-level encoder-decoder modeling method for question answering (QA) from large-scale knowledge bases (KB). This method improves the existing approach [9] from three aspects. First, long short-term memory (LSTM) structures are adopted to replace the convolutional neural networks (CNN) for encoding the candidate entities and predicates. Second, a new strategy of generating negative samples for model training is adopted. Third, a data augmentation strategy is applied to increase the size of the training set by generating factoid questions using another trained encoder-decoder model. Experimental results on the SimpleQuestions dataset and the Freebase5M KB demonstrates the effectiveness of the proposed method, which improves the state-of-the-art accuracy from 70.3% to 78.8% when augmenting the training set with 70,000 generated triple-question pairs.

Keywords: Question Answering, Knowledge Base, Long Short-Term Memory, Encoder-Decoder

1 Introduction

As the scale of structured knowledge bases (KB) grows, how to take full advantage of them gets more and more attention. One of the popular research topics is knowledge base-based question answering (KB-QA), which aims to answer natural language factoid questions using the triples in knowledge bases. Developing a high-accuracy KB-QA system has a lot of applications, such as the next generation searching engines, digital assistants, and so on.

The existing approaches to KB-QA can be summarized into three main categories. The first is the semantic parsing-based approach [16, 15, 11, 7, 1, 14]. This approach usually constructs a semantic parser to translate natural language queries into structured expressions, i.e., logic-forms, and then derives answers from large-scale KBs using these generated query expressions. The second is the information retrieval-based approach [13]. This approach usually uses the information conveyed in questions to search answers from KBs, and adopts ranking techniques to make final selection among candidate answers. The third is the vector space modeling-based approach [6, 3]. This approach maps both natural

language questions and all triples in KBs into low-dimensional embedding vectors. An input question is answered by finding the triple in the KB which has the highest similarity score with the question in the embedding vector space. At training time, the model parameters are estimated to maximize the similarity scores for the question-answer pairs in the training set.

With the rapid development of deep learning techniques in natural language processing, some researchers have started to introduce deep structured neural networks into the vector space modeling-based approach to KB-QA in recent years. Various neural network architectures, such as memory networks [4] and convolutional neural networks (CNN) [8], have been employed to derive the vector representations for questions and triples and to measure the similarity scores between them. A character-level encoder-decoder modeling method for KB-QA was proposed [9]. In this method, character-level encoders were adopted to deal with the data sparsity issue of using word-level encoders. Long short-term memory (LSTM) models [10] and convolutional neural networks were applied to encode input questions and triple elements (i.e., entities and predicates) respectively. An LSTM model with attention was built to decode the optimal entities and predicates according to the encoding results. All model parameters were estimated in an end-to-end manner using a training set of question-triple pairs. This method achieved the state-of-the-art performance on the SimpleQuestions dataset without use of ensembles.

This paper improves this character-level encoder-decoder modeling method [9] from three aspects. First, LSTMs are adopted to replace CNNs for encoding the candidate entities and predicates. Second, a new strategy of generating negative samples for model training is adopted. Third, inspired by the recently proposed encoder-decoder-based question generation method [12], a data augmentation strategy is applied to increase the size of the training set by generating factoid questions from KB triples to further alleviate the data sparsity issue. Experimental results on the SimpleQuestions dataset [4] with the Freebase KB [2] demonstrates the effectiveness of our proposed method. Before data augmentation, this method achieves an accuracy of 77.5% in the Simplequestions setting, outperforming previous state-of-the-art accuracy of 70.3% [9] by 7.2%. Furthermore, an accuracy of 78.8% is obtained when augmenting the training set with 70,000 generated triple-question pairs.

2 Character-Level Attention Model with LSTM Encoders

This paper works on single-relation question answering. Thus, the aim of the character-level attention model is to decode every natural language question into an entity and a predicate, which can uniquely determine a triple in the KB. Let q denote the input question, $\{e\} = e_1, ..., e_N$ and $\{p\} = p_1, ..., p_M$ denote a set of candidate entities and predicates respectively. This model calculates the probability of $p(e_i, p_j|q)$ for each $i \in 1...N$ and $j \in 1...M$.

The model structure is shown in Fig 1. Three character-level LSTM encoders are adopted to transform question texts, entity names and predicate names into

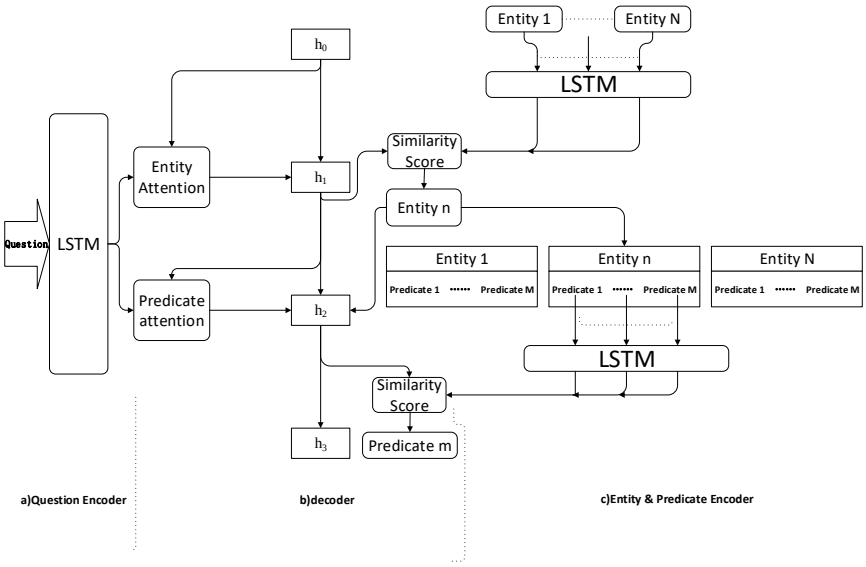

Fig. 1. The model structure of question answering with character-level LSTM encoders.

embedding vectors. An LSTM decoder with attention mechanism is employed to calculate the similarity scores between the input question and candidate triple elements. These similarity scores are used to calculate $p(e_i, p_j|q)$ in order to find the most likely (entity, predicate) pair for question answering. This model structure is very similar to the one proposed in Golub's work [9]. The difference is that LSTMs are adopted to replace the CNNs for encoding the names of candidate entities and predicates. Compared with CNNs, LSTMs are expected to be more capable of sequence modeling. The details of each component in Fig 1 are briefly described in the following subsections.

2.1 Encoding the Question, Entity and Predicate

Two steps are taken to encode the question, entity and predicate. First, three groups of one-hot encoding vectors are extracted to represent each character in question texts, entity names and predicate names respectively. Then, three LSTM encoders are built to accept these character-level encoding vectors as input. When encoding questions, we keep the outputs at all time steps and get a sequence of embedding vector for each input question. When encoding entities and predicates, we choose the output vector at the last time step or conduct average pooling along time axis to get the embedding vectors.

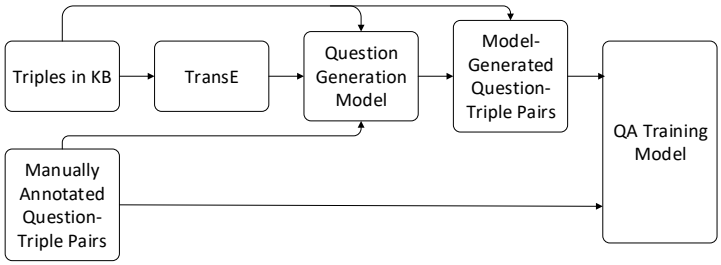

Fig. 2. The flowchart of data augmentation with model-based question generation.

2.2 Decoding the KB Query

The decoder aims to get the single entity and predicate for deriving the right answer to the input question. As shown in Fig 1, the entity and the predicate are decoded in two steps separately. An LSTM with attention mechanism is built and the hidden states at each step are used to decode the most likely entity and predicate. A pairwise semantic relevance function [9] is employed to measure the similarity between the hidden states of LSTM and the embedding vectors of candidate entities and predicates. More detailed introduction to the attention-based LSTM and the semantic relevance function can be found in [9].

3 Data Augmentation with Model-based Question Generation

The performance of neural network-based KB-QA methods are always constrained by the amount of available question-answer or question-triple pairs for model training. Recently, an encoder-decoder-based question generation method was proposed [12]. This method considered the mapping from a triple in KBs to a natural language question as a translating process and adopted an encoder-decoder framework to achieve it. The encoder transformed each triple into a vector using embedding matrices pre-trained by TransE [5]. In TransE, the predicate of a triple (*topic entity, predicate, answer entity*) in the KB is considered as as a transformation between the topic entity and the answer entity. The objective function of TransE training is to make the sum of the topic entity vector and the predicate vector close to the answer entity vector. The estimated TransE model can easily transform each triple in the KB into a vector as its output. Then, the vector of the output of TransE was fed into an LSTM-decoder to generate a natural language question. All model parameters were estimated using human annotated question-triple pairs. It was reported that this method can generate questions indistinguishable from real human-generated ones [12].

Inspired by this question generation method, this paper presents a data augmentation strategy to increase the size of the training set by generating factoid

questions from KB triples and to alleviate the data sparsity issue for QA model training.

The flowchart of this strategy is shown in Fig 2. Given a training set with human-annotated question-triple pairs and a large-scale KB for KB-QA, we first train a TransE model to get the embedding matrices for all entities and predicates in the KB. Then, An encoder-decoder-based question generation model is built using the pre-trained TransE model and the human-annotated question-triple pairs following the method proposed in [12]. Finally, a large amount of questions can be produced using the question generation model and the triples in the KB. These model-generated questions are combined with the human-annotated ones for training the QA model introduced in Section 2.

4 Experiments

4.1 Experimental Conditions

We evaluated our proposed method on the SimpleQuestions dataset and the Freebase5M KB [4]. The original dataset consist of 108,442 single-relation questions and their corresponding triples formed as (*topic entity, predicate, answer entity*). It is usually split into 75,910 question-triple pairs for training, 10,845 pairs for validation, and the remaining 21,687 pairs for test. In our implementation, we removed the pairs whose topic entity can not find a name string in the Freebase5M KB. Therefore, we finally got 75,519 training samples, 10,787 validation samples, and 21,573 test samples respectively.

For an input question, we took the entities in the Freebase5M KB whose name matched an n-gram substring of the question as candidate entities. Simple statistics showed that the number of matched entity names for all questions in the SimpleQuestions dataset was less than 7. Thus, we fixed the number of candidate entity names to 7 and added some candidate entity names randomly for the questions whose matched entity names were less than 7. For each candidate entity name, the entity in the Freebase5M whose name was identical to this candidate entity name were added to the set of candidate entity. If the number of entity matching a candidate entity name was larger than 10, we sorted these entity by the number of facts they had in the KB and the top-10 entity were added to the set of candidate entity. For each candidate entity, the predicates in the triples whose topic entity was in these candidate entity were appended to the set of candidate predicates. We fixed the number of candidate predicates to 150 for each candidate entity name and also added candidate predicates randomly for these entity names with less than 150 linked predicates. Finally, the number of candidate pairs of (topic entity name, predicate) for each question was 7×150.

When building our character-level attention model with LSTM encoders, the character-level encoding vectors were 200-dimensional and the three LSTM encoders for questions, entities, and predicates all had one hidden layer of size 200. When encoding entities and predicates, we either chose the output vector at the last time step or calculated the average of the outputs at all time steps as

the embedding vectors. The LSTM decoder also had a hidden layer of size 200. The model parameters were estimated using AdaDelta with the learning rate of 0.0001.

4.2 Comparison on Negative Sample Generation Methods

In Golub's work [9], the candidate entities and predicates for model training both consisted of a true answer and 50 randomly sampled answers. In our implementation, we adopted the candidate generation process introduced in Section 4.1 to produce the negative samples for model training. We compared the performance of using Golub's method and the proposed candidate generation method for producing negative samples. The results are shown in Table 1, from which we can see that the proposed candidate generation method achieved an accuracy of 76.7% and outperformed the random generation method by 5.11%.

Negative Sample Generation	Joint Acc.	Entity Acc.	Predicate Acc.
Golub's method [9]	71.59	91.76	71.68
Proposed method	76.70	91.80	76.79

Table 1. QA accuracy (%) of using different negative sample generation methods for model training.

4.3 Comparison on Pooling Methods of LSTM Encoders for Entities and Predicates

In our proposed model structure shown in Fig 1, the LSTM encoders for entities and predicates are required to produce a single vector representation for each entity or predicate. Since the raw outputs of LSTMs are sequential, a pooling procedure is necessary. In this experiment, we compared the performance of using the output vector at the last time step or the average of all output vectors as the encoding results. The results are shown in Table 2. From this table, we can see that using average pooling achieved a better accuracy than using the last vector. This is reasonable because the averaged vector may convey more global information of the text string than the last vector given by LSTM encoders. Thus, this average pooling strategy were adopted in the following experiments.

Pooling Methods	Joint Acc.	Entity Acc.	Predicate Acc.
Last	76.70	91.80	76.79
Average	77.50	92.00	77.56

Table 2. QA accuracy (%) of using different pooling ways of LSTM encoders for questions,entities and predicates.

4.4 Effects of Data Augmentation

We built two augmented training sets for comparison. The *T_set* was composed of the original training set of SimpleQuestions and another 70,000 questions generated using a fixed template as "*What is the P of E ?*", where E denoted the entity name in a triple and P meant the predicate [4]. The *M_set* consisted of the original training set of SimpleQuestions and 70,000 questions produced by the encoder-decoder-based question generation method introduced in Section 3. Two models were built to achieve this model-based data augmentation.

1. The first one was a TransE [5] model as showed in Fig 2. Due to the sparsity of triples in the SimpleQuestions training set, an augmented KB based on the SimpleQuestions training set and the Freebase5M KB was built for TransE training. Simple statistics showed that there were 7,523 predicates in Freebase5M while only 1,629 predicates in SimpleQuestions training set. We built an intermediate set by extracting those triples in Freebase5M whose predicates were in the SimpleQuesitons training set and totally got 16,561,736 triples. The final augmented KB for TransE training was composed of the triples in Freebase5M whose topic entities were in the intermediate set. There were 36,291,331 triples in the final augmented KB and the output KB embeddings given by TransE had 200 dimensions.
2. The second one was the encoder-decoder model for question generation built following the method proposed in [12]. The encoder part accepted the KB embeddings produced by the TransE model as inputs and generated a 600-dimensional representation vector for each input triple. Then, this vector was fed into the decoder part, which was a GRU-recurrent neural network (GRU-RNN) with attention. The hidden layer of the GRU-RNN had 600 units. The Simplequestions training set was used to train this encoder-decoder with a learning rate of 2.5×10^{-4}.

Both sets approximately doubled the original training set of SimpleQuestions. Two character-level attention models for KB-QA were built using the two augmented training sets and the results are shown in Table 3. It can be observed that the data augmentation strategy was helpful and the model-based question generation method achieved more performance improvement than the conventional template-based method.

Training Set	Joint Acc.	Entity Acc.	Predicate Acc.
SimpleQuestions	77.50	92.00	77.56
T_set	77.91	91.94	77.99
M_set	78.81	92.29	78.85

Table 3. QA accuracy (%) of data augmentation.

4.5 Comparison with other existing methods

We compared the performance of our proposed methods and some existing methods. The results are shown in Table4. Both methods (1) and (2) adopted memory networks [4] for KB-QA and built models at word-level. Method (2) used ensembles of multiple models and combined the WebQuestion training set and a paraphrase dataset to deal with the data-sparsity issue. The difference between our proposed method and Method (3) [9] in Table 4 has been discussed before. From this table, we can see that our proposed method achieved an accuracy of 77.5% in the Simplequestions setting without data augmentation, which outperformed other existing methods listed in Table 4. Furthermore, an accuracy of 78.8% was obtained when augmenting the training set with 70,000 generated triple-question pairs.

Method	Joint Accuracy
(1) MenNN [4]	61.6
(2) MemNN-Ensemble [4]	63.9
(3) Character Attention [9]	70.9
(4) proposed method without data argumentation	77.5
(5) proposed method with data argumentation	78.8

Table 4. QA accuracy (%) of proposed methods and some existing methods.

4.6 Analysis and Discussion

Comparison between using LSTMs or CNNs to encode entities and predicates We compared the performance of using LSTMs or CNNs to encode entities and predicates in our implementation. The results are shown in Table 5. Here, the CNN had two alternating convolutional and fully-connected layers, followed by one fully-connected layer. The width of filters and the number of feature maps in convolution layers were set to 4 and 100 respectively. All the fully-connected layers had 200 output units. The other modules of the two systems were the same. Form this table, we can see the effectiveness of encoding entities and predicates using LSTMs.

Model	Joint Acc.	Entity Acc.	Predicate Acc.
CNN	73.1	91.7	73.2
LSTM	77.5	92.0	77.6

Table 5. QA accuracy (%) of using LSTMs or CNNs to encode entities and predicates.

Discussion on negative sample generation method As introduced in Section 4.1, randomly selected entities and predicates were used during candidate generation in order to achieve fixed numbers of candidate entities and predicates for each question. An experiment was also conducted to remove these randomly selected candidates for model testing, and the results are shown in Table 6. From this table, we can see that randomly adding candidates helped to achieve a better performance of question answering.

Add Random Candidates	Joint Acc.	Entity Acc.	Predicate Acc.
Yes	78.81	92.29	78.85
No	72.11	92.74	72.38

Table 6. QA accuracy (%) with or without adding random candidates.

We made some further analysis to investigate the reason of the performance difference in Table 6. There were totally 21,573 questions in our test set. When using random candidate entities and predicates, there were 1,266 test questions whose target entity can not be found in the candidate entities and there were 480 test questions whose target entities were in the candidate set but predicates not. Without adding random candidates, these two numbers were 1,266 and 2,801 respectively. The number increase from 480 to 2,801 indicates the advantage of adding random candidates, which is to construct a candidate set with better coverage on the target predicates of test questions. We also tried to remove the test questions whose target entities or predicates were missing in the candidate sets and re-evaluated the two testing set in Table 6. The results are shown in Table 7. Comparing Table 6 with Table 7, we can see that the performance of both systems got improved when the candidate sets can provide an 100% coverage of the correct ones. In Table 7, the QA accuracy of using random candidates is lower than the one without using random candidates. This means that adding random candidates increases the difficulty of model inference when all correct answers are in the candidate set. Therefore, there exists a trade-off between the coverage of candidate sets and the difficulty of model inference in our implementation.

Add Random Candidates	Joint Acc.	Entity Acc.	Predicate Acc.
Yes	85.75	98.04	85.79
No	88.86	98.52	89.20

Table 7. QA accuracy (%) with or without adding random candidates. The test questions whose target entities or predicates were missing in the candidate sets were removed.

5 Conclusion

This paper has proposed a new character-level encoder-decoder modeling method for simple question answering. We have improved the existing approach [9] by employing LSTMs to encode entities and predicates, introducing a new strategy to generate negative samples for model training, and augmenting training set with neural-network-based question generation method. Our proposed method has achieved a new state-of-the-art accuracy of 78.8% on the SimpleQuestions dataset and the Freebase5M KB. To investigate better candidate generation strategy, to build larger augmented training set and to combine the advantages of word-level and character-level modeling will be the tasks of our future work.

Acknowledgements

This paper was supported in part by the National Natural Science Foundation of China (Grants No. U1636201) and the Fundamental Research Funds for the Central Universities (Grant No.WK2350000001).

References

1. Berant, J., Chou, A., Frostig, R., Liang, P.: Semantic parsing on freebase from question-answer pairs. In: EMNLP. vol. 2, p. 6 (2013)
2. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: 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 (2008)
3. Bordes, A., Chopra, S., Weston, J.: Question answering with subgraph embeddings. arXiv preprint arXiv:1406.3676 (2014)
4. Bordes, A., Usunier, N., Chopra, S., Weston, J.: Large-scale simple question answering with memory networks. arXiv preprint arXiv:1506.02075 (2015)
5. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems. pp. 2787–2795 (2013)
6. Bordes, A., Weston, J., Usunier, N.: Open question answering with weakly supervised embedding models. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 165–180. Springer (2014)
7. Cai, Q., Yates, A.: Large-scale semantic parsing via schema matching and lexicon extension. In: ACL(1). pp. 423–433 (2013)
8. Dong, L., Wei, F., Zhou, M., Xu, K.: Question answering over freebase with multi-column convolutional neural networks. In: ACL (1). pp. 260–269 (2015)
9. Golub, D., He, X.: Character-level question answering with attention. arXiv preprint arXiv:1604.00727 (2016)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9(8), 1735–1780 (1997)
11. Kwiatkowski, T., Choi, E., Artzi, Y., Zettlemoyer, L.: Scaling semantic parsers with on-the-fly ontology matching. In: In Proceedings of EMNLP. Percy. Citeseer (2013)

12. Serban, I.V., García-Durán, A., Gulcehre, C., Ahn, S., Chandar, S., Courville, A., Bengio, Y.: Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. arXiv preprint arXiv:1603.06807 (2016)
13. Yao, X., Van Durme, B.: Information extraction over structured data: Question answering with freebase. In: ACL (1). pp. 956–966. Citeseer (2014)
14. Yih, S.W.t., Chang, M.W., He, X., Gao, J.: Semantic parsing via staged query graph generation: Question answering with knowledge base (2015)
15. Zettlemoyer, L.S., Collins, M.: Learning context-dependent mappings from sentences to logical form. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2. pp. 976–984. Association for Computational Linguistics (2009)
16. Zettlemoyer, L.S., Collins, M.: Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. arXiv preprint arXiv:1207.1420 (2012)