A Fine-grained Complex Question Translation for KBQA

Guangxi Ji, Shujun Wang, Ding Zhang, Xiaowang Zhang*, and Zhiyong Feng

College of Intelligence and Computing, Tianjin University, Tianjin 300350, China
{guangxiji, shujunwang, zhangdingTJU, xiaowangzhang, zyfeng}@tju.edu.cn

Abstract. Translating natural language questions into SPARQL queries is a significant challenge of semantic parsing based KBQA due to the gap between their representations. In this paper, we designed a fine-grained complex question answering framework for KBQA, including a semantic similarity model and a neural machine translation model. Based on the above two models, we present a complex question processing algorithm to transform questions into subqueries and then process them parallelly. The experiments evaluated on benchmark datasets show that our approach is significantly effective.

Keywords: Question Answering · Question Decomposition · Semantic Textual Similarity · Neural Machine Translation

1 Introduction

Knowledge Base Question/Answering(KBQA) system can automatically answer questions asked in natural language over the knowledge base. A widely used approach is to translate natural language questions into SPARQL queries so that the question can be answered by executing its corresponding query. However, it has become a challenge as there is a gap in their representation. The existing methods based on semantic parsing or templates require a large number of high-quality rules or templates constructed manually or automatically. The matching restrictions based on strings and structures are relatively strict [1][2]. Other methods using neural machine translation models fail to identify and link unseen entities to corresponding knowledge base entities [3].

In this paper, we propose a semantic similarity model to decompose a complex question into several simple subquestions to achieve fine-grained translation. Here we find the question pattern similar to subquestion on the semantic level. Next, we translate these subquestions in parallel using our neural machine translation(NMT) model and finally assemble the subqueries to obtain the corresponding complete SPARQL query.

2 Approach

Our approach for complex question answering, shown in Fig.1, can be divided into 5 steps.

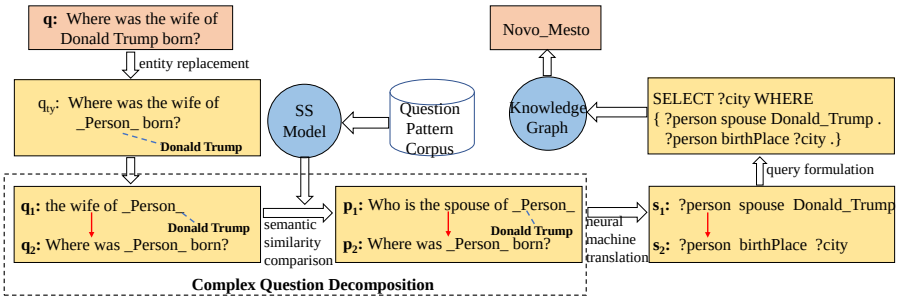

Fig. 1. An example of complex question process.

Step(1) Named Entity Replacement. We focus on the structure information of natural language questions. Specifically, we use the named entity linking tools to identify and replace the entities in the question with their corresponding entity classes in the knowledge base to get the question pattern, which represents a kind of question.

Step(2) Semantic Similarity-based Question Decomposition. The decomposed subquestions may be incomplete, that is, some components are missing. It may be wrong to translate them directly into SPARQL queries. Therefore, we present a semantic similarity model based on the siamese network architecture to find the most semantically similar standard question. Specifically, given a question q, we transform it into a fixed-size embedding by adding a pooling layer after BERT. Having these embeddings, we can use cosine-similarity to calculate the semantic similarity between questions. Finally, we use mean squared-error loss as the objective function, making the semantic similarity question closer.

Similar to Zheng et al[4], the underlying principle of our decomposition algorithm is to try each subquestion of question q and complete its similarity with patterns in T, where T is our question pattern corpus. Algorithm 1 presents the detail of our question decomposition algorithm. Lines 5 to 8 deal with parallel complex questions, while lines 9 to 18 show the decomposition method of nested complex questions.

Step(3) Neural Machine Translation. Translation methods based on templates or rules require accurate matching, while neural machine translation has better generalization. We use the Transformer-based neural machine translation model, which is mainly composed of two parts: question & query encoding and translation. The former models the semantics of question and query into the embedding representation so that the transformer model could transfer semantic information between different expressions. Here, we follow the encoding

approach suggested by Soru et al.[8]. Note that, unlike Sour et al.[8], which input a complete natural language question and output the corresponding SPARQL query, our model inputs the simple question pattern and output its corresponding triple pattern, which effectively solves the linking problem of entities that have never appeared before and improves the accuracy of translation.

Step(4) Query Construction. After parsing all simple subquestions of a complex question, we need to assemble their corresponding triple patterns together to form a complete SPARQL query to obtain the answer. Algorithm 1 shows that the decomposition of complex questions is orderly. The first subquestion can be operated independently, and the others need to use the previous results as part of its facts. Here we assemble all the triple patterns into a complete query in the order of decomposition. The variable of the last pattern is taken as the variable of the SPARQL query. Note that we need to unify the join variables for triple patterns that have join relationships and replace the entity class with the real entity.

Step(5) Query Evaluation. Evaluating the query to get the final answer.

Algorithm 1: QD(q_{ty}, T_{en}, M, τ)

Input: Question pattern q_{ty} =$\{w_1, \cdots, w_n\}$, Encoded pattern set T_{en}, Semantic Similarity model M and the similarity threshold τ;
Output: The decomposed subquestion patterns P(q)

$q_{en} \leftarrow$ M.encode(q_{ty})
$(\gamma, t) \leftarrow$ the maximum similarity between q_{en} and T_{en}
if $\gamma \geq \tau$ **then**
    return $P(q) \leftarrow t$
if *"and","or","but", etc. in* q_{ty} **then**
    for q_{sub} *in GetSubQuestion(*q_{ty}*)* **do**
        $P(q) \leftarrow P(q) \cup$ QD(q_{sub}, T_{en}, M, τ)
    return $P(q)$
for $i \in$ *[1,*$|q_{ty}|$*]* **do**
    $e_i \leftarrow$ the position of the first entity class after w_i
    for $k \in$ *[*e_i*,*$|q_{ty}|$*]* **do**
        $q_{sub} \leftarrow$ GetSubstring(q_{ty},i,k)
        $q_{en'} \leftarrow$ M.encode(q_{sub})
        $(\gamma, t) \leftarrow$ the maximum similarity between $q_{en'}$ and T_{en}
        if $\gamma \geq \tau$ **then**
            $q_{ty'} \leftarrow$ replace q_{sub} in q_{ty} with the answer type of t
            if $|q_{ty'}|$*=1 or QD(*$q_{ty'}$*,* T_{en}*, M,* τ*)* $\neq$ *NULL* **then**
                return $P(q) \leftarrow P(q) \cup t$
return NULL

3 Experiments and Results

The evaluation of our method is performed on three datasets(i.e., LC-QuAD[5], QALD-9[6], and ComplexQuestions[2]), using F1 measure as the metric. We use a large number of simple questions to construct training data. Specifically, our training data consists of two parts: the SimpleDBpediaQA[9] dataset and some common simple questions collected from WikiAnswers. The former is a benchmark dataset for simple question answering over knowledge base, which contains 43086 questions and the entities contained in each question. WikiAnswers[10] is a large corpus of natural languae questions. For each question, the corresponding question pattern can be obtained through Step(1).

For the semantic similarity model, we construct many training data in the form of $\{p_1, p_2, s\}$, where p_1 and p_2 are two question pattern, here p_2 may be a transformation of p_1 (such as changing structure, omitting compoents, replacing synonyms), the similarity score s is obtained by considering their structure, words, semantic, etc. For instance, {"who is the spouse of _Person_","the wife of _Person_",1.0 }. Besides, using the existing KBQA system, we can obtain the sparql query corresponding to each question, and filter out the correct (question pattern, query pattern) pairs as training data for the neural machine translation model.

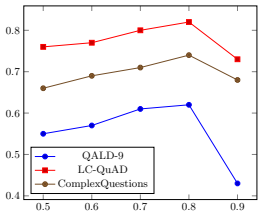

Fig. 2. Effect of the threshold τ

	F1 Score
Bast and Haussmann++ [7]	0.47
QUINT [2]	0.49
TemplateQA [4]	0.71
Our Approach	0.74

Table 1. Results on ComplexQuestions

As shown in Table 1, our method achieves better results because we decompose the question at the semantic level and can obtain each subquestion and its standard form more accurately. Our method is based on the DBpedia knowledge base, and others have not been tested on the QALD-9 and LC-QuAD datasets, so there is no comparison here. From the data shown in Fig 2, our method works best on the LC-QuAD dataset because templates generate it, so the question structure is similar and can capture subqueries better. The QALD-9 dataset is more complex, and its effect is somewhat different from the others. We also study the influence of semantic similarity threshold τ. When τ is small, some subsequences are mistakenly considered simple subquestions, resulting in the wrong decomposition. While τ is large, some subquestions can not find the corresponding question pattern, so the transformation fails. A large number of experiments show that 0.8 is a better threshold.

4 Conclusion

In this paper, we propose a method to decompose complex questions into multiple simple subquestions to achieve fine-grained translation using a semantic similarity model. Here we find the question pattern similar to subquestion on the semantic level. In addition, we translate these subquestions in parallel using the neural machine translation model. We hope that our work can inspire other applications of deep learning methods in KBQA.

5 Acknowledgments

This work is supported by the National Key Research and Development Program of China (2017YFC0908401) and the National Natural Science Foundation of China (61972455). Xiaowang Zhang is supported by the Peiyang Young Scholars in Tianjin University (2019XRX-0032).

References

1. W. Zheng and M. Zhang, "Question Answering over Knowledge Graphs via Structural Query Patterns", *arXiv:1910.09760*, 2019.
2. A. Abujabal, M. Yahya, M. Riedewald, and G. Weikum, "Automated template generation for question answering over knowledge graphs", in *Proceedings of the 26th International Conference on World Wide Web (WWW)*,2017, pp. 1191–1200.
3. X. Yin, D. Gormann and S. Rudolph, "Neural Machine Translating from Natural Language to SPARQL", in *arXiv:1906.09302*, 2019.
4. W. Zheng, J. X. Yu, L. Zou, and H. Cheng, "Question answering over knowledge graphs: Question understanding via template decomposition", in *Proceedings of the VLDB Endowment*, vol. 11, no. 11, pp. 1373–1386, 2018.
5. P. Trivedi, G. Maheshwari, M. Dubey and J. Lehmann, "LC-QuAD: A Corpus for Complex Question Answering over Knowledge Graphs", in *16th International Semantic Web Conference*, pp. 210–218, 2017.
6. R. Usbeck, R. H. Gusmita, A. C. N. Ngomo and M. Saleem, "9th Challenge on Question Answering over Linked Data (QALD-9)", in *17th International Semantic Web Conference*, pp. 58–64, 2018.
7. H. Bast, E. Haussmann,"Proceedings of the 24th ACM International on Conference on Information and Knowledge Management," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015, pp.1431–1440.
8. T. Soru, E. Marx, D. Moussallem and G. Publio, A. Valdestilhas, D. Esteves, C. Baron Neto, "SPARQL as a Foreign Language," in *SEMANTiCS*, 2017.
9. M. Azmy, P. Shi, J. Lin and I. F. Ilyas, "Farewell Freebase: Migrating the SimpleQuestions Dataset to DBpedia," in *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 2093–2103, 2018.
10. A. Fader, L. S. Zettlemoyer, and O. Etzioni. "Paraphrase-Driven Learning for Open Question Answering", in *ACL*, 2013.