Multiple Question Answer Optimization in RDF Q/A System*

Abstract. RDF Question/Answer(Q/A) system allows users to ask questions in natural languages over Knowledge Graph represented by RDF. To answer a natural language question, a lot of excellent work has been done. However, existing work on RDF Q/A mainly focuses on a-question-at-a-time approaches. When multiple questions arrive at the same time, sequential processing is not always the most efficient. In this paper, We first propose a novel approach for efficiently detecting useful semantic information in natural language. Then we give a way to improve the efficiency of the RDF Q/A system by using the common semantic parts. Experiment confirm that our method not only keeps the precision but also speeds up query performance greatly.

Keywords: Multiple-Question · Knowledge Graph · RDF Q/A.

1 Introduction

Knowledge Graph(KG) has become a very popular way to represent and query world knowledge. RDF and SPARQL are two core concepts in the Knowledge Graph, which are standards for representing and querying Knowledge Graph, recommended by the W3C. Although SPARQL is a standard way to access RDF data, it remains tedious and difficult for users. Hence, qustion/answering(Q/A) based on Knowledge Graph has received wide attention in both natural language processing and database areas.

Generally, there are two significant challenges in RDF Q/A systems:***question understanding***(Question → SPARQL) and ***query evaluation***. A great deal of research has been done to address the first challenge. For example, Zou et al[2]. proposed two frameworks to build the semantic query graph. To overcome the second challenge, single-machine RDF systems, like RDF-3X[3],gStore[5] and many distributed SPARQL query engines have been introduced.

However, in some scenarios multiple queries can be processed as a batch. For example, in Q/A system, multiple questions often need to be processed together. Therefore, the third challenge arises in RDF Q/A system: "How can RDF Q/A System handle batch questions efficiently". Some research work has been proposed to efficiently process multiple SPARQL queries[4]. However, efficient processing of multiple SPARQL queries is not equivalent to multiple Question Answer. Because batch processing of SPARQL queries is only part of the second challenge.

The main contribution of this paper is an effective solution to the multiple questions answer optimization in RDF Q/A system.

* Supported by organization x.

2 Overview of Our Approach

The main contribution of this paper is an effective solution to the multiple Question Answer optimization problem in RDF Q/A System. Specifically,

1. By searching and utilizing the common structure in the Question undestanding stage, the system efficiency is improved.
2. How to measure the core degree of words in a natual language question for generating SPARQL queries is given.

2.1 Detecting Common Structure

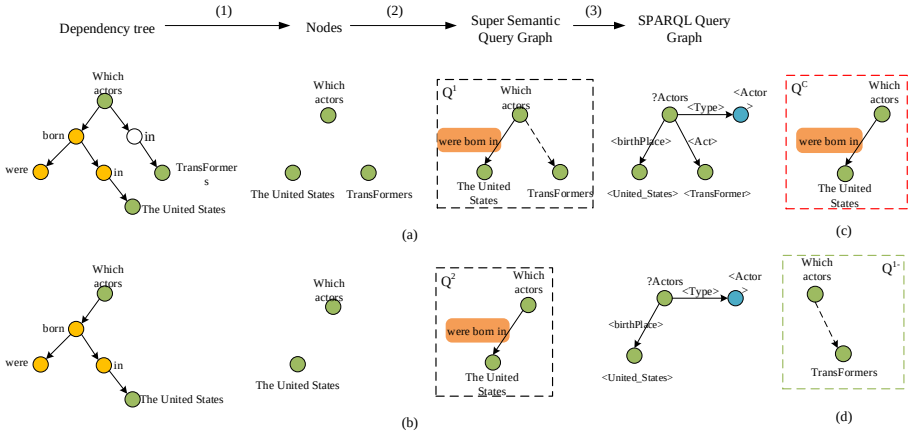

Fig. 1. Acceleration Ratio

Figure 1 (a) shows a process of transforming question "*Which actors in Transformers were born in the United States?*" into a SPARQL query graph. The first step is to recognize all nodes from the question sentence N.(1) Generally, we extract entities, classes and wild-cards as nodes. We adopt the dictionary-based entity linking approach [5] to find entities and classes. Given a node set V (which has been recognized in the first step) and a dependency tree Y of question sentence, for any two nodes v_i and $v_j (\in V)$, we introduce an edge between v_i and v_j if and only if the simple path between v_i and v_j does not contain other node in V. Finally, the phrases linking operation is executed.

Assuming that there are two questions N_1("*Which actors in Transformers were born in the United States?*") and N_2("*Which actors were born in the United States?*") to be processed by RDF Q/A system, Figure 1 (a) shows the process of question N_1 and Figure 1 (b) shows the process of N_2. We can see that there is a common structure Q^C(Figure 1(c)) between Super Semantic Query Graph

generated in (a) and (b). We can take this common part out to execute Phrases Linking separately, and then question N_1 only needs to execute Phrases Linking operation on Q_1^-(Figure 1(d)) part.

2.2 Semantic Capture

In the challenge of Question Understanding, we tried to use the SPARQL Q to accurately express the semantics of the question N. In fact, the different words in N are different in the importance of generating Q. However, there is currently no way to measure the importance of each word in N for generating Q. Question Understanding can be expressed by the following formula:$f(N) \to Q$.

The natural language question N is composed of the word w_i, and the SPARQL is composed of triple pattern t_j, so that we can convert the $f(N) \to Q$ into $f(w_1, w_2, ..., w_n) = (t_1, t_2, ...t_m)$, and then by vectorizing w_i and t_j we can get the following formula:

$$f(\overrightarrow{w_1}, \overrightarrow{w_2}, ..., \overrightarrow{w_n}) \to (\overrightarrow{t_1}, \overrightarrow{t_2}, ..., \overrightarrow{t_m}) \tag{1}$$

we defined a loss function as follows.

$$L = \sum(\sum_{i=1}^{m} \overrightarrow{t_i} - \sum_{j=1}^{n} \overrightarrow{w_i}) \tag{2}$$

By Equation 3, we make the overall $\langle N_i, Q_j \rangle$ in the data set as equal as possible. That is, $(\overrightarrow{w_1} + \overrightarrow{w_2} + ... + \overrightarrow{w_n}) = (\overrightarrow{t_1} + \overrightarrow{t_2} + ... + \overrightarrow{t_m})$. So we give each w_i a suitable vector representation. Then we can use the following formula to measure the core of a word w_i in N.

$$\omega = \frac{\sum_{j=1}^{m} w_i^j}{\sum_{i=1}^{n} \sum_{j=1}^{m} w_i^j} \tag{3}$$

Where w_i^j represents the value of the vector w_i in the jth dimension.

With the method of getting the core degree of each word w_i in N, we need to consider how to get a large enough $\langle Question, Query \rangle$ data set, that is, how to get a SPARQL Query Q corresponding to Natual Language N correctly. We chose to use gAnswer to generate a SPARQL Q for each question in the $\langle Question, Answer \rangle$ dataset(SQuAD). If Q returns the same result as Answer on the data set, the $\langle Question, Query \rangle$ pair is retained. Then we get a large enough $\langle Question, Query \rangle$ data set.

3 Experiment and Evaluation

From the data shown in Table 1, we can see that the accuracy of Our Approach is slightly higher than the original paper gAnswer because of the better selection of words w_i in natural language problem N (filtering out some possible interference information).

Table 1. Evaluating QALD Testing Questions.

	Processed	Right	Recall	Precision	F-1
Our Approach	100	70	0.74	0.90	0.80
gAnswer	100	68	0.72	0.89	0.78
RFF	100	40	0.43	0.77	0.55
KWGAnswer	100	52	0.59	0.85	0.70
Aqqu	100	36	0.37	0.39	0.38

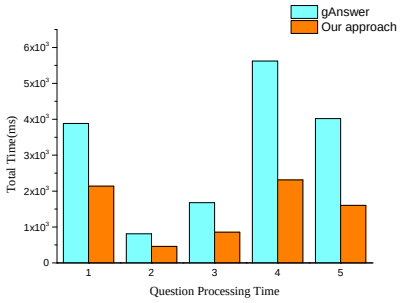

Fig. 2. Question processing time

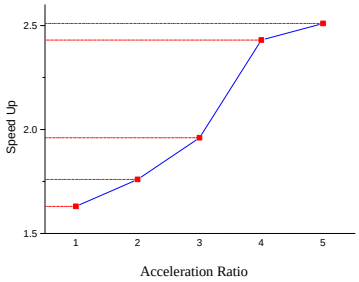

Fig. 3. Acceleration Ratio

Figure 2 shows that by using our method, the RDF Q/A system greatly improves the efficiency of batch query processing with common structures. As shown in Figure 3, the efficiency is improved between 1.5 and 2.5 times. The rate of efficiency increase depends on the proportion of the entire structure occupied by the public structure. The greater the proportion of common structures, the more efficient it is.

References

1. James Bailey, Alistair Moffat, Charu C. Aggarwal, Maarten de Rijke, Ravi Kumar, Vanessa Murdock, Timos K. Sellis, and Jeffrey Xu Yu, editors. *CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*. ACM, 2015.
2. Sen Hu, Lei Zou, Jeffrey Xu Yu, Haixun Wang, and Dongyan Zhao. Answering natural language questions by subgraph matching over knowledge graphs (extended abstract). In *ICDE 2018, Paris, France, April 16-19, 2018*, pages 1815–1816, 2018.
3. Thomas Neumann and Gerhard Weikum. RDF-3X: a risc-style engine for RDF. *PVLDB*.
4. Xuguang Ren and Junhu Wang. Multi-query optimization for subgraph isomorphism search. *PVLDB*, 10(3):121–132, 2016.
5. Lei Zou, M. Tamer Özsu, Lei Chen, Xuchuan Shen, Ruizhe Huang, and Dongyan Zhao. gstore: a graph-based SPARQL query engine. *VLDB J.*, 23(4):565–590, 2014.