CASIA@QALD-3: A Question Answering System over Linked Data

Shizhu He, Shulin Liu, Yubo Chen,
Guangyou Zhou, Kang Liu, and Jun Zhao

National Laboratory of Pattern Recognition,
Institute of Automation, Chinese Academy of Sciences.
Zhongguancun East Road 95#, Beijing, 100084, China.
{shizhu.he,shulin.liu,yubo.chen,gyzhou,kliu,jzhao}@nlpr.ia.ac.cn

Abstract. We present a question answering system (CASIA) over Linked Data (DBpedia), which focuses on construct a bridge between the users and the Linked Data. Based on the Linked Data consisting of subject-property-object (SPO) triples, each natural language question firstly is transformed into a triple-based representation (Query Triple). Then, the corresponding resources in DBpedia, including class, entity, property, are mapped for the phrases in the query triples. Finally, the optimal SPARQL query is generated as the output result. Our system can not only deal with the single-relation questions but the complex questions containing multi-relations. We evaluate our approach on QALD-3 test dataset and achieve an F-measure score of 0.36, an average precision of 0.35 and an average recall of 0.36 over 99 questions.

Keywords: Question Answering, Linked Data, DBpedia

1 Introduction

With the rapid development of the Web of Data, there are many RDF datasets published as Linked Data[1]. Thus, developing user-friendly interface for accessing those linked data become increasing important. However, there are the "gaps" between the users and the Linked Data. On the one hand, the users, even expert programmers, need a lot of practices to handle standard structured query languages like SPARQL. On the other hand, due to the diversity and high heterogeneity of the Linked Data, it is difficult for humans to select relevant resources and discover useful information.

Question answering over Linked Data is aimed at eliminating those "gaps", which attempts to allow the users to access those structured data with natural language. To fulfill this aim, some systems have already been proposed during the last few years. PARALEX[2] maps open-domain questions to subject-property-object (SPO) triples

database by leveraging an automatically constructed lexicon. The limitation of [2] is that it can only answer the simple question which only contains one single relation, but fail to deal with the questions with complex structures. DEANNA[3] believes the translating natural language questions into the structured SPARQL queries consists with several disambiguation tasks. They solve those disambiguation tasks jointly by using an integer linear program. However, some questions still cannot be answered by both systems, such as aggregation, filter, comparative and superlative questions. TBSL[4] addresses those problems by utilizing some hand-written templates, and filling slot in SPARQL templates with resources (entity, class or property).

In this paper, we demonstrate our system in QALD-3, which focuses on translating natural language expression into standard RDF query language expression (SPARQL). Our system implements a basic pipeline framework which consists three main components, including question analysis, resource mapping and SPARQL generation. In specific, we first employ shallow and deep linguistic analysis to transform NL-queries into a set of Query Triples with <subject, predict, object> format. Second, we map each phrase in Query Triple to the corresponding resource (class, entity, or property) in DBpedia. As a result, Ontology Triples are generated. Thirdly, the SPARQL queries will be constructed based on Ontology Triple and question type. At last, the generated SPARQL queries is used to search on the Linked Data, and the best answer can be picked out through validating and ranking. The framework of our system is similar to PowerAqua[5]. However, besides different technologies used for question analysis, the big difference is that we use relational patterns when mapping relations rather than use WordNet and the context to disambiguate the phrase in query triples. Furthermore, our system is similar to QAKiS[6], which is to exploit a relation-based matching and construct relational patterns automatically from Wikipedia. In contrast, our work focuses not only the simple questions, but also on complex questions. Moreover, we use several free resources including PATTY[7] and WordNet[8] instead of constructing relation pattern repository by ourselves.

2 System Description

The architecture of our system will be presented in this section. We will give a detailed description of each component and give a step by step explanation with the following example:

Who are the parents of the wife of Juan Carlos I?(id=67, test set).

2.1 System Architecture

The current version of our QA system targets the questions which do not contain comparative and superlative words. We also do not consider the questions which contain aggregation and filter condition. Generally, our system is composed of three main steps (as shown in Figure 1):

1) Question Analysis: Given a natural language question, we first detect the type of this question (yes/no, number, etc.) and extract the name entities from it. Then we

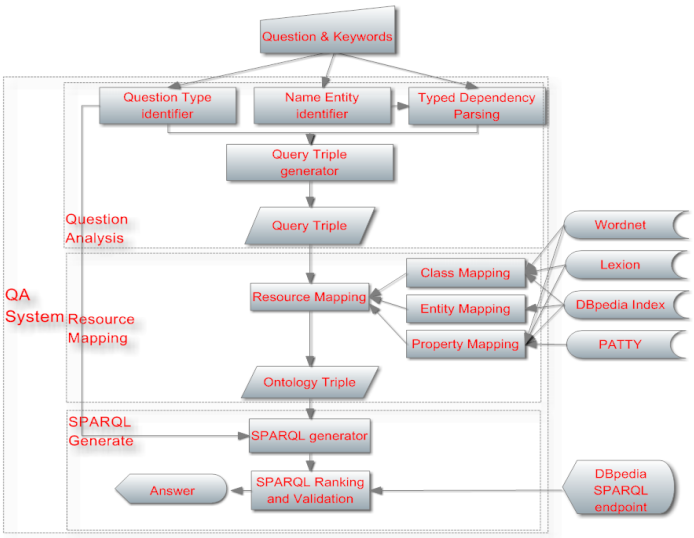

Fig. 1. The architecture of CASIA

modify some phrases and their POS tags by combining name entities and keywords. After that, we take deep linguistic analysis to generate the typed dependency parsing tree for this question. At last, a set of linguistic triples <subject, predicate, object> is constructed from the dependency parsing tree. The upper example will be transformed to two query triples, where the second one is a sub-triple of the first one:

<?who, are the parents of, wife>
<?wife, the wife of, Juan Carlos I >

2) Resource Mapping: For every phrase in query triple, the most important thing is to identity the corresponding resource in Linked Data, such as DBpedia. For different types of resource, we use different techniques and resources. The output of this step is a list of ontology triples. One query triple will generate several possible ontology triples. Possible triples of the aforementioned example are:

<?Person, rdf:type, dbc:Person>
<?Person, dbp:parent, ?wife>
<?wife, dbo:spouse, dbr:Juan_Carlos_I_Of_Spain>

3) SPARQL Generation: Based on identified question type (bool, number, etc.) and the results of resource mapping, we generate SPARQL query candidates. Then DBpedia SPARQL endpoint will be used to validate those query candidates. The legal queries with the highest score will be selected and returned as the right answer. If there are several different results with the same highest score, we combine them using SPARQL language keyword “UNION”. A SPARQL query candidate of the example question is as following:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/resource/>
SELECT DISTINCT ?URL
WHERE {
      ?URL rdf:type dbo:Person.
      ?URL dbo:parent ?wife.
      ?wife dbo:spouse dbp:Juan_Carlos_I_of_Spain.
}
```

2.2 Question Analysis

We design some simple heuristics to identify the question type, based on which we will use different SPARQL templates to generate queries. For example, the yes/no question will use "**ask where**" template; the "**sum**" question will use "**select count(?x)**" template. In general, we use the following regular expressions:

— "**^ (are|did|is|was|does|were|do).***", corresponding to the yes/no question;

— "**^ how(many|tall|long).***", corresponding to the "sum" question;

—"**^ (when|how often|since when|how long|for how long).***", corresponding to the Date question;

— "**^ (what|which|where|who).***" and "**^ (show me|give me|list).***", corresponding to normal "select" question.

For a yes/no question, we transform it to a declarative sentence based on phrase structure parsing tree. For example, *"Is Michelle Obama the wife of Barack Obama?" (id=67, test set)* will be transformed to *"Michelle Obama is the wife of Barack Obama."*.

Considering the Linked Data consisting subject-property-object (SPO) triples, we transform questions into query triples based on dependency parsing tree. Given a natural language question and corresponding keywords, we use Stanford NER tool[1] to identify name entities in question, and then generate a typed dependency parsing tree combining keywords, name entities and POS tags. However, the name entities in question cannot be occasionally identified by Stanford NER tool. Through our coarse statistics, there are 74 entities identity correctly in all 95 entities from 99 questions. However, we cannot extract any NEs from such question *"Give me the Apollo 14 astronauts"*; and we just extract *"Juan Carlos"* from question *"Who are the parents of the wife of Juan Carlos I?"* (the right result is *"Juan Carlos I"*). Therefore, we use keywords of question provide by QALD-3 organizers; we extract keywords with capitalization as proper nouns. For example, the parsing result of the example question is showed as follows(Figure 2): Then, we use some heuristics to construct query triple from parsing tree. For example, we collect the phrases in the path "nsubj→prep→ pobj" and transform it into a triple. If there is a dependency path in a phrase ("*the wife of Juan Carlos I*" in the "pobj" of phrase "*parents*"), we will construct an additional query triple. Because we mainly extract subject-predicate-object and proposition phrases relations, we cannot extract query triples for many questions, such as the question *"Which films starring Clint Eastwood did he direct himself?"*.

[1] `http://nlp.stanford.edu/software/corenlp.shtml`

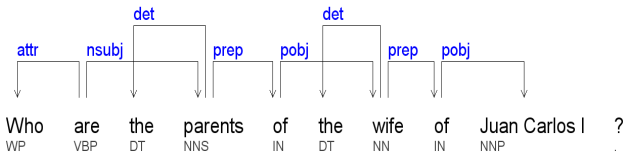

Fig. 2. The dependency parsing tree of the example question

2.3 Resource Mapping

Query triples are the intermediate representation form of user's natural language question. To generate SPARQL queries, the most important thing is mapping them to the right "resource" in Linked Data. There are four type resources in Linked Data: Class (concept), Instance (Entity), Property (Relation) and Literal. Currently we ignore the Literal resource. Thus, the task of this step is to find appropriate item in those three resources sets by given a phrase in query triple.

We assume that different positions of a query triple will be mapped to different type resources. Thus, in <Subject, Relation, Object>, "Subject" and "Object" may be mapped to Class and Instance, and "Relation" will be mapped to Property. To fulfilling this mapping, we build three Lucene[2] Indexes separately. However, finding different types of resource needs different techniques. In specific, mapping to instance, index lookup in the inverted index can yield good performance. Additionally, we can find the right resources even given different spellings by applying Wikipedia redirects. For example, any phrase of *"MJ23"*, *"His Airness"*, *"Micheal Jordan"* and *"Michael Jeffery Jordan"* will be mapped to the same URI < `http://dbpedia.org/resource/Michael_Jordan`>. Sometimes the phrase may contain a class, such as in the question *"Which books by Kerouac were published by Viking Press?"*, *"book"* should be mapped to the class URI < `http://dbpedia.org/ontology/Book`>. If the complete match cannot find appropriate URIs for phrase, we will perform the following steps to revise the results:

1) Search the similar resources and calculate their string similarities (Levenshtein distance) with the given words;

2) expand the given words using WordNet;

3) using lexical word when encountering "prep", such as *"by"* will invoke *"dbo:author"*.

In the current version of our system, we just invoke four relations using hand-crafted rules, including *"by"* invokes *"dbo:starring"* and *"dbo:author"*; *"in"* and *"of"* invoke *"dbo:country"*, *"dbo:starring"* and *"dbo:locationCity"*.

However, when mapping property, we may still fail to get right properties through the above steps. For example in the question *"Who is the husband of Amanda Palmer?"*, *"husband"* should be mapped to *"dbo:spouse"* with URI < `http://dbpedia.org/ontology/spouse`>. Several recent studies focus on finding the relation patterns for a property, such as BOA[9] and PATTY[6]. We use PATTY[3] instead of collect the Relational Patterns by ourselves.

[2] `http://lucene.apache.org`

[3] `http://www.mpi-inf.mpg.de/yago-naga/patty/`

2.4 SPARQL Generation

Each phrase in query triple may be mapped to more than one resources in the Linked Data. To filter excessive resources, we set a similarity threshold to rank SPARQL queries. For every query triple S,P,O, we generate S,P,?x and ?x,P,O in order to check whether the subject or object actually occurs with the property. If one question has 3 conditions (3 query triples), we will generate $2^3 = 8$ SPARQL queries. Based on the identified question type, we generate complete SPARQL query, and use DBpedia SPARQL endpoint[4] to validate the legality of each candidate query. The process of validating and constructing rules is presented as follows:

—bool question: For the "bool" type questions, we use "ask" template. The query is legal only if it returns "true", and we cannot deal with the question which answer is "wrong".

—number question: At first we use "select" template. And, if the retrieval results are more than one, or the only one result is not an integer value, we will use "select count" template.

—other question: If the query could get results, it is a legal query.

Each legal query will be associated with a score. We select the queries with the highest score, and combine them using "UNION" if there are more than one queries sharing the same highest score.

3 Results on QALD-3 data

Table 1 gives the evaluation results with average precision, average recall and F-measure. It shows the number of question our system can answer, the number of right and partially right answers among them.

Table.1. Evaluate results on QALD-3 test dataset.

Total	Right	Partially right	Avg.Precision	Avg.Recall	F1
99	29	8	0.35	0.36	0.36

Table 2 gives the time consumption of 10 questions selected randomly. Except the question ID, and the questions shown in column 1 and column 2 respectively, column 3 shows the number of generated query triples generated, column 4 shows question analysis step time consumption (second), and the SPARQL generated step time consumption (second) is shown in column 5. The average time consumption of all 99 QALD-3 test set questions is shown in the last row. The more query triples generated in the first two steps, the more time cost in the final step. Yet, the time cost in question analysis for different questions is almost the same. The "OUT OF SCOPE" question do not have final step.

[4] `http://vtentacle.techfak.uni-bielefeld.de:443/sparql`

Table.2. Time consumption on answering QALD-3 test questions.

ID	Question	Number of Query Triples	Time in first two steps	Time in first last step
2	Who was the successor of John F. Kennedy?	18	0.07	28.223
19	Give me all people that were born in Vienna and died in Berlin.	10	0.459	8.684
20	How tall is Michael Jordan?	39	3.083	35.177
21	What is the capital of Canada?	5	0.141	4.249
54	What are the nicknames of San Francisco?	11	0.122	9.324
67	Who are the parents of the wife of Juan Carlos I?	240	7.566	855.661
81	Which books by Kerouac were published by Viking Press?	336	8.479	410.263
86	What is the largest city in Australia?	22	6.886	29.646
97	Who painted The Storm on the Sea of Galilee?	18	2.887	30.016
100	Who produces Orangina?	NULL	13	NULL
	Average on all 99 questions	60.54	3.28764	83.2855

3.1 Error Analysis and Discussion

As introduced before, our current version system cannot address questions which contain comparative, superlatives and negation words. We cannot answer complex questions with relative sentences and conjunctions also. For example, we cannot address question *"Which films starring Clint Eastwood did he direct himself?" (id=88, test set)*

In our observation, the mistakes in our system can be classified into four types. We discuss each of the four type errors in detailes.

First, some errors occur in the question analysis step. Because we just extract subject-predicate-object and proposition phrases relations, wrong query triples generated for some questions. For example, we only extract <?people, born in, Vienna> in question *"Give me all people that were born in Vienna and died in Berlin."(id = 19, test set)*, and the proper query triples should have two coordination triples <?people, born in, Vienna> and <?people, died in, Berlin >. We can deal with the questions with cascade relation. For example, in question *"Which books by Kerouac were published by Viking Press?"(id = 81, test set)*, *"publish by"* is modify the phrase *"books by Kerouac"*.

Second, there are some entities mapping errors. We do not resolve the entity disambiguation problem, some entities extracted from question are wrong using string matching. For example, we extract entity *"GMT"* (dbp:GMT) in question *"List all games by GMT."(id = 40, test set)*, but *"GMT"* was an ambiguities word and it corresponded *"GMT Games"* (dbp:GMT_Games) in this question.

Third, considering the ontology we rely on for concept extraction, some concepts (class) CASIA cannot identify. For example, *"surfers"* cannot be identified as proper concept in question *"Which professional surfers were born on the Philippines?"(id = 46, test set)*.

Finally, most of our current version systems mistakes concern wrong relation mapping. For example, we cannot extract any relation in question *"Which countries are*

connected by the Rhine?"(id = 45, test set), the same situation occur in *"What is the ruling party in Lisbon?"(id=53, test set)* and *"How tall is Michael Jordan?"(id = 20, test set)*. Even though PATTY was a good relation patterns repository, mapping its patterns to relation properties in DBpedia was not perfect.

Except upper errors, our system also cannot detect the right order between subject and object. For example, in question *"Who are the parents of the wife of Juan Carlos I?"(id = 67, test set)*, we generated SPARQL expression contain meanings of *"Who are the parents of the wife of Juan Carlos I?"* and *"Who are the child of the wife of Juan Carlos I?"*. The same problem occurs in the question *"Did Socrates influence Aristotle?"(id = 62, test set)*.

4 Conclusion and Future Work

In this paper, we present a question answering system over Linked Data, which translate the natural language questions into the standard RDF data queries (SPARQL). The pipeline of our system consists of three main components, including question analysis, resource mapping and SPARQL generation. In specific, we use shallow (POS tagging, NER) and deep linguistic analysis (dependency parsing) to transform each natural language question into triple-based representation. Then we use PATTY, WordNet and other free external resources to find the corresponding resource of the phrases in the query triples. At last, based on the identified question type, we generate the SPARQL queries and use the DBpedia endpoint to validate them. However, we still cannot answer the comparative and superlative questions. In addition, the linguistic analysis cannot identify all relations in questions. In the future we will address those problems and include more Linked Data (currently we only use DBpedia).

5 Acknowledgements

Thanks to Prof. Hang Li, Dr. Yibo Zhang and Dr. Jie Zhang at Huawei Noah'S Ark Lab for their insightful advices. This work was supported by the National Natural Science Foundation of China (No. 61070106, No. 61272332 and No. 61202329), the National High Technology Development 863 Program of China (No. 2012AA011102), the National Basic Research Program of China (No. 2012CB316300).

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Journal on Semantic Web and Information Systems (in press) (2009)
2. Fader, A., Zettlemoyer, L., Etzioni, O.: Paraphrase-Driven Learning for Open Question Answering. In: Proc. of ACL'13(2013)
3. Yahya, M., Berberich, K., Elbassuoni, S.,Ramanath, M.,Tresp, V., Weikum, G.: Natural Language Questions for the Web of Data. In: Proc. of EMNLP'12(2012)
4. Unger, C., Bhmann, L., Lehmann, J., Ngomo, A., Gerber, D., Cimiano, P..: Template-based Question Answering over RDF Data. In: Proc. of WWW'12(2012)

5. Lopez, V., Fernndez, M., Stieler, N., Motta, E.: PowerAqua: supporting users in querying and exploring the Semantic Web content. Semantic Web Journal, In Press (2011)
6. Cabrio, E., Aprosio, A., Cojan, J., Magnini, B., Gandon, F., Lavelli, A.: QAKiS @ QALD-2. In: Proceedings of Interacting with Linked Data (ILD 2012) [37] (2012), 87-95. http://ceur-ws.org/Vol-913/07-ILD2012.pdf
7. Nakashole, N., Weikum, G., Suchanek, F.: PATTY: A Taxonomy of Relational Patterns with Semantic Types. In: Proc. of EMNLP'12(2012)
8. George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41
9. Gerber, D., Ngomo, A.: Bootstrapping the Linked Data Web. 1st Workshop on Web Scale Knowledge Extraction, ISWC'11(2011)