A Study of URI Spotting for Question Answering over Linked Data (QALD)

KyungTae Lim(✉), NamKyoo Kang, and Min-Woo Park

Korea Institute of Science and Technology Information, Daejeon, Korea
{kyungtaelim,ngkang,pminwoo}@kisti.re.kr

Abstract. Recently, there have been studies on linked data-based question answering systems such as Question Answering over Linked Data (QALD). In these linked data-based question answering systems, it is essential to connect URI which is used as an identifier of the linked data with a chunk or surface form constituting such queries. This study attempts to suggest effective URI spotting that connects question-answering chunk and URI of the linked data using conventional natural language processing techniques. In addition, this study addresses a solution for selecting best linked data entities from chunk by using automatically generated standard RDF query language expression (SPARQL) queries.

Keywords: Question answering over linked data · Question answering · URI spotting · Linked data

1 Introduction

Conventional question answering systems have been dependent upon natural language processing technology in many parts. However, answers can differ by reflecting current situations. For example, an answer to the question "Who is the President of the Republic of Korea?" can change every five years. In other words, database should be updated periodically in a conventional question answering system. In fact, a semantic web technology-based question answering system can be a good backup plan. In question answering systems such as QALD [1–3], answer to the queries is found from DBpedia which is updated on a regular basis.

For the QALD, a key step titled 'Resource Mapping' is the most important step [4]. In this study, it called URI Spotting because 'Resource Mapping' seems to be broad mapping. URI spotting refers to a process of finding the resources of the associated linked data from queries as shown in Fig. 1 below. In the QALD, URI spotting should be precisely completed to prepare SPARQL queries which return answers. Therefore, it is a very important process. According to conventional studies, in the QALD system which assumed that URI spotting was accurately completed, F1-score recorded 0.9 [5] which is about 3 times higher than the average (0.32) of the teams where analysis was performed without considering URI spotting results. Hence, a high level of URI spotting can reveal high question-answering accuracy. This study proposes a method to

DOI: 10.1007/978-3-319-48390-0_17

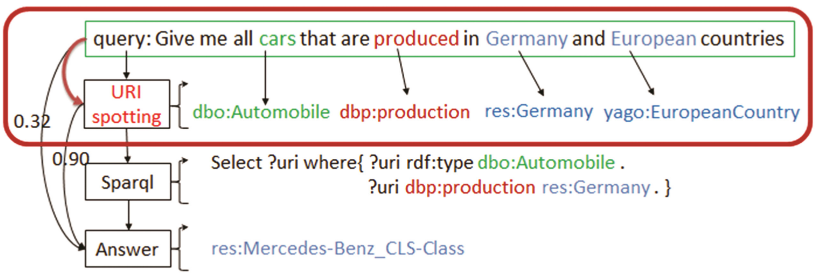

query: Give me all cars that are produced in Germany and European countries
URI spotting
dbo:Automobile
dbp:production
res:Germany
yago:EuropeanCountry
0.32
0.90
Sparql
Select ?uri where{ ?uri rdf:type dbo:Automobile .
?uri dbp:production res:Germany . }
Answer
res:Mercedes-Benz_CLS-Class

Fig. 1. Example (in Red) of URI spotting and FI-scores in each step for QALD (Color figure online)

effectively promote URI spotting by using basic language processing techniques such as morphological analysis and named entity recognition. Based on the techniques, it is able to find entities such as the subjects and objects for solving question-answers.

2 URI Spotting for QALD

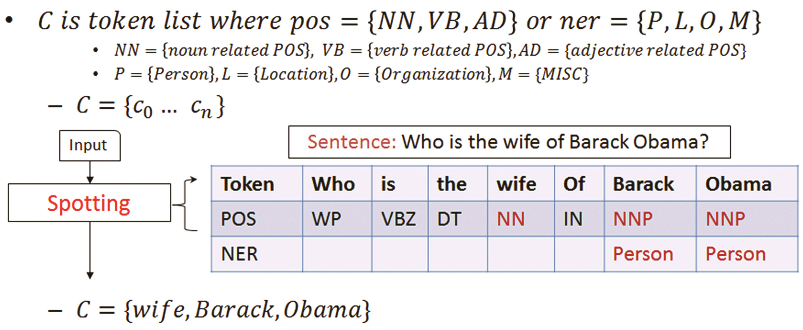
URI spotting refers to a process of finding the resources of the associated linked data from queries. In this study, URI spotting can be divided into three processes:

- Spotting: Extracts candidates for URI spotting using the related techniques such as morphological analysis and NER;
- Candidate extension: Extends candidates using similar algorithm, WordNet synset and other identifiers based on the candidates extracted during the spotting stage;
- Selection: Selects the best candidates by combining SPARQL queries from the resources extracted during the candidate extension.

2.1 Spotting

Spotting is the first process for extracting URIs. It is a process to select the subjects to be mapped with the linked data resources from the words constituting sentences. It includes an entity boundary detection process, which has the same meaning of the terms designed in DBpedia Spotlight [6]. In DBpedia spotlight, however, spotting is conducted against nouns. In URI spotting, in contrast, nouns and verbs are targeted for spotting because inter-entity property information is needed to handle question answering. The property information of QALD often comes from verbs so that it has a different scope for spotting from the DBpedia spotlight which targets entities only. Therefore, nouns and verbs are targeted for spotting in terms of part-of-speech tagging.

Figure 2 reveals the spotting procedure and example of actual sentence processing. In initial formula, ‘C’ refers to a data structure in a list form constructed for spotting. As stated above, the subjects for spotting are extracted against tokens with various parts of speech such as noun, verb and adjective and those with the results of recognizing named entities. In this example, ‘is’ was ignored. In the results of ‘C,’ therefore, wife, Barack and Obama become the subjects for spotting.



- C *is token list where* $pos = \{NN, VB, AD\}$ *or* $ner = \{P, L, O, M\}$
  - $NN = \{noun\ related\ POS\},\ VB = \{verb\ related\ POS\}, AD = \{adjective\ related\ POS\}$
  - $P = \{Person\}, L = \{Location\}, O = \{Organization\}, M = \{MISC\}$
- $C = \{c_0 \dots c_n\}$

Sentence: Who is the wife of Barack Obama?

Token	Who	is	the	wife	Of	Barack	Obama
POS	WP	VBZ	DT	NN	IN	NNP	NNP
NER						Person	Person

- $C = \{wife, Barack, Obama\}$

Fig. 2. Procedure and example of spotting, the 1st step of URI spotting

2.2 Candidate Extension

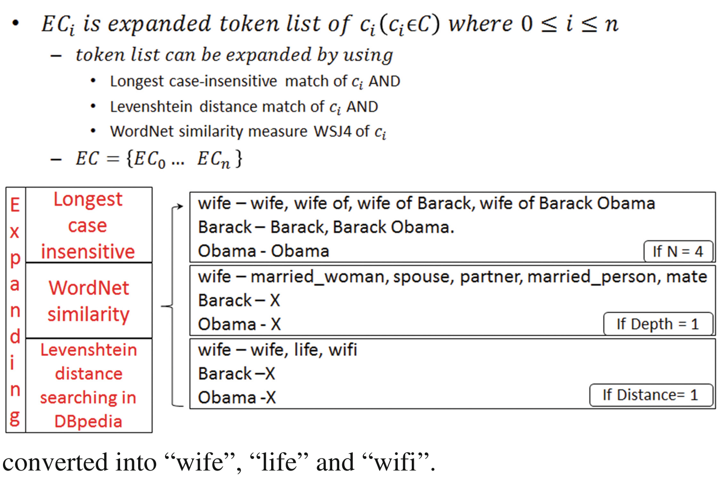
The resource of DBpedia could be a single word or mixture of several words. For example, A question "Who is the wife of Barack Obama?" can be analyzed as follows:

- Input: Who is the wife of Barack Obama?
- URI spotting output: dbo:spouse, res:Barack_Obama

In this example, the nouns (wife, Barack and Obama) become the candidates for spotting based on the spotting algorithm mentioned above. However, it is hard to find resources in DBpedia with the candidates for spotting only. Therefore, candidate extension algorithm which extends candidates from the spotted words is needed. The candidate extension algorithm is divided into as follows:

- Similar string extension algorithm: Levenshtein distance-based algorithm
  e.g., the word 'wife': wife, wifi, fife
- Long string choice algorithm: Longest case-insensitive match algorithm
  e.g., the word 'wife': wife of Barack Obama, wife of Barack, wife of, wife
- Wordnet similarity measurement algorithm: Algorithm using synonym dictionary, wordnet and others
  e.g., the word 'wife': wife, spouse, married to

Figure 3 introduce the extension algorithms and examples. In the Fig. 3's formula, 'EC' refers to a list of extended candidates. It is able to extend candidates in three algorithms. First, in similar string extension algorithm, DBpedia resource words having 'N' or lower in the distance of two strings are found using Levenshtein distance. This algorithm was very helpful in finding property. In the sentences which start with "Who wrote," in particular, it revealed great results along with lemmatization. In the queries which begin with "Who wrote," it is needed to find the property "writer." Therefore, it is able to get "write" as the lemmatization result of "wrote" and "write", "wrote" and "writer" using the Levenshtein distance. The example at the bottom in Fig. 3 shows the Levenshtein distance extension of the word "wife." When the distance was set to '1' in DBpedia, the extensible DBpedia resources can be converted into "wife", "life" and "wifi".



- EC_i *is expanded token list of* $c_i (c_i \epsilon C)$ *where* $0 \leq i \leq n$
  - *token list can be expanded by using*
    - Longest case-insensitive match of c_i AND
    - Levenshtein distance match of c_i AND
    - WordNet similarity measure WSJ4 of c_i
  - $EC = \{EC_0 \ldots EC_n\}$

Expanding
Longest case insensitive
WordNet similarity
Levenshtein distance searching in DBpedia
wife – wife, wife of, wife of Barack, wife of Barack Obama
Barack – Barack, Barack Obama.
Obama - Obama
If N = 4
wife – married_woman, spouse, partner, married_person, mate
Barack – X
Obama - X
If Depth = 1
wife – wife, life, wifi
Barack –X
Obama -X
If Distance= 1
converted into "wife", "life" and "wifi".

Fig. 3. Extended candidate set according to candidate extension algorithm

In the long string choice algorithm, string is created through extension up to the word behind the preset scope from "wife." The first example in Fig. 3 reveals an attempt of extension having the extension coefficient as '4.' It was designed to get and select bigger weight values as string becomes longer when there are several DBpedia resources in a single spotting at the selection of optimum candidates.

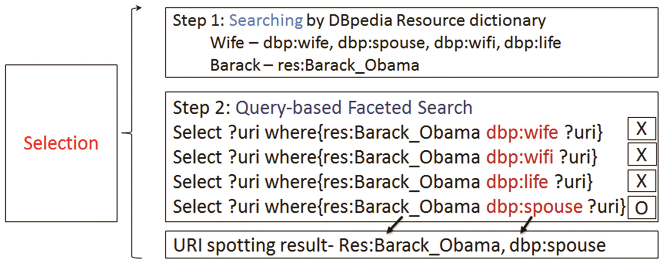
In Wordnet similarity measurement algorithm, the distance score algorithm for Wordnet's synset hierarchical structure was used. In case of a middle example in Fig. 3, when the depth of hierarchical structure was set to '1,' it reveals the hypernyms of "wife": "spouse", "married_woman", "partner", "married_person" and "mate." In this study, Wordnet synset whose hierarchical depth of hyponyms, hypernyms and synonyms is '1' was adopted as shown in the example. If the hierarchy becomes deeper, accuracy can rather decrease because of increase in the number of candidates. In a set of the extended candidates in Fig. 3, consequently, all results converted from the three algorithms should be added.

2.3 URI Selection

The single-spotted chunk through candidate extension algorithm owns at least one extended set. However, a testing step is needed because URI spotting requires a selection of optimum URI to solve QALD questions. The URI selection algorithm consists of two steps: First, it is needed to examine if there are extended candidates in DBpedia by searching the DBpedia resource dictionary. For example, if there are extended candidates (wife", "wifi", "spouse", "wife of" and "wife of Barack") in the single-spotted chunk "wife," each extended candidate is searched and returned in DBpedia resource dictionary. In this case, "dbp:wife", "dbp:wifi", "dbp:life" and "dbo:spouse" are returned.

The 2nd step is to return the optimum candidate. For example, as shown in Step 2 in Fig. 4, results are selected depending on if query results are returned by combining SPARQL queries. Figure 5 shows the SPARQL creation & selection algorithm. As stated above, the URI section algorithm is the 1st step which examines if there are the candidates extended from DBpedia resource dictionary and stores those under inspection only. In the algorithm in Fig. 5, ‘C’ refers to the candidates from the single-spotted unit while ‘E’ represents the candidates returned to entities through dictionary search among the spotted candidates during the 1st step (URI selection algorithm). When there is one extended candidate from the spotted unit, the optimum candidate selection algorithm selects it. If there are several candidates, both ‘E’ and ‘C’ were returned to optimum candidates if SPARQL is normally operated after going through a SPARQL combination test with those identified as entities.

- EC *is divided into* $entityEC_x$ *and* $propertyEC_y$
- *If* `select ?uri where {entityECx, propertyECy, ?uri}` *where* $0 \le x \le l$ *and* $0 \le y \le m$ *and* $l + m = n - 1$ returns ∅ then disregard it
- *Otherwise, add it into answer candidate list.*
- *Return answer candidate list as a result.*



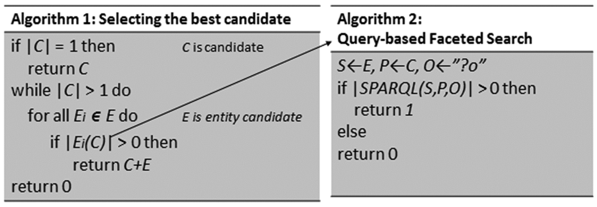
Step 1: Searching by DBpedia Resource dictionary
Wife – dbp:wife, dbp:spouse, dbp:wifi, dbp:life
Barack – res:Barack_Obama

Step 2: Query-based Faceted Search

Select ?uri where{res:Barack_Obama dbp:wife ?uri}	X
Select ?uri where{res:Barack_Obama dbp:wifi ?uri}	X
Select ?uri where{res:Barack_Obama dbp:life ?uri}	X
Select ?uri where{res:Barack_Obama dbp:spouse ?uri}	O

URI spotting result- Res:Barack_Obama, dbp:spouse

Fig. 4. Finding optimum set by combining SPARQL queries from the extended candidates



```
Algorithm 1: Selecting the best candidate
if |C| = 1 then                C is candidate
   return C
while |C| > 1 do
   for all Ei ∈ E do           E is entity candidate
      if |Ei(C)| > 0 then
         return C+E
return 0
```

```
Algorithm 2:
Query-based Faceted Search
S←E, P←C, O←"?o"
if |SPARQL(S,P,O)| >0 then
   return 1
else
return 0
```

Fig. 5. SPARQL-based candidate selection algorithm

The candidate selection algorithm creates SPARQL queries based on entity (E) candidates and checks if they function properly [7]. For example, if there are “res:Barack_Obama” in ‘E’ and “spouse” and “wife” in ‘C,’ SPARQL queries are created as stated in Table 1.

Among the four SPARQL candidates, the SPARQL to be returned is “res:Barack_Obama dbo:spouse?who,” and the return results are “res:Michel_Obama,” which is an answer to actual queries. In this case, therefore, the optimum candidate selection algorithm is “dbo:spouse, res:Barack_Obama”.

Table 1. Creation of SPARQL queries using candidate selection algorithm

Subject	Predicate	Object
res:Barack_Obama	dbo:spouse	?who
?who	dbo:spouse	res:Barack_Obama
res:Barack_Obama	dbo:wife	?who
?who	dbo:wife	res:Barack_Obama

3 Conclusion and Future Work

The purpose of the study was to construct a URI spotting system especially for solving QALD questions. For this, a lot of natural language processing techniques were utilized. For further works, we will test our system to prove it is effective to solve answering for QALD questions. And also try to analysis relations what kind of natural language processing techniques lead batter result for URI spotting.

Acknowledgements. This research was supported by the Sharing and Diffusion of National R&D Outcome funded by the Korea Institute of Science and Technology Information

References

1. Shekarpour, S., Ngomo, A.C.N., Auer, S.: Question answering on interlinked data. In: Proceedings of the 22nd international conference on World Wide Web, pp. 1145–1156. ACM, (2013)
2. Unger, C., Forascu, C., Lopez, V., Ngomo, A.C.N., Cabrio, E., Cimiano, P., Walter, S.: Question answering over linked data (QALD-4). In: Working Notes for CLEF 2014 Conference (2014)
3. Cimiano, P., Lopez, V., Unger, C., Cabrio, E., Ngonga Ngomo, A.-C., Walter, S.: Multilingual question answering over linked data (QALD-3): lab overview. In: Forner, P., Müller, H., Paredes, R., Rosso, P., Stein, B. (eds.) CLEF 2013. LNCS, vol. 8138, pp. 321–332. Springer, Heidelberg (2013)
4. He, S., Liu, S., Chen, Y., Zhou, G., Liu, K., Zhao, J.: CASIA@ QALD-3: a question answering system over linked data. In: Proceedings of Multilingual Question Answering over Linked Data (QALD-3), Workshop co-located with CLEF, Valencia (2013)
5. Ferré, S.: squall2sparql: a Translator from Controlled English to Full SPARQL 1.1. In: Work. Multilingual Question Answering over Linked Data (QALD-3) (2013)
6. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: DBpedia spotlight: shedding light on the web of documents. In: Proceedings of the 7th International Conference on Semantic Systems, pp. 1–8. ACM, New York (2011)
7. Guyonvarch, J., Ferre, S., Ducassé, M.: Scalable query-based faceted search on top of SPARQL endpoints for guided and expressive semantic search. In: Proceedings of the Question Answering over Linked Data lab. Research report (2013)