Question Answering with DBpedia Based on the Dependency Parser and Entity-centric Index

Huiying Li
School of Computer Science and Engineering
Southeast University
Nanjing, P.R. China
huiyingli@seu.edu.cn

Feifei Xu
School of Computer Science and Engineering
Southeast University
Nanjing, P.R. China
xufeisea@126.com

***Abstract*—The emerging Linked Open Data provides an opportunity to answer the natural language question based on knowledge bases (KB). This study proposes an approach to question answering (QA) on the DBpedia dataset. After parsing the question by a dependency parser, we locate the entity mention and property mention with predefined templates. We propose an entity-centric indexing model to help search referent entities in KB. After obtaining the referent entities, we expand the property mention with WordNet and ConceptNet to find the referent properties of the returned entities. The values of the referent property are then considered the answer to the question. Evaluations are performed on DBpedia version 2015. Results show that our approach reaches 46% precision when the top-10 entities are returned in the final QA stage. The evaluation tests show that our approach is promising in dealing with QA in Linked Data.**

Keywords- Linked Open Data; RDF; Question answering; Dependency parser

I. INTRODUCTION

Linked Open Data (LOD) aims to publish structured data to enable the interlinking of such data and therefore enhance their utility. It shares information that can be read automatically by computers and allows data from different sources to be connected and queried. LOD consists of an unprecedented volume of structured datasets, such as DBpedia and Freebase, currently amounts to 50 billion facts that are represented as Resource Description Framework (RDF) triples on the web.

The large amount of Linked Data has become an important resource to support question answering. However, many challenges are encountered in returning a right answer based on the knowledge base for a natural language question (utterance). The following example represents a question and some snippets in DBpedia.

Sample question: "what town was martin luther king assassinated in?"

Sample knowledge base: A snippet of DBpedia dataset is shown in Fig. 1, which lists an instance with its type, supertype, name, spouse, cause of death, place of death, year of death, and place of birth.

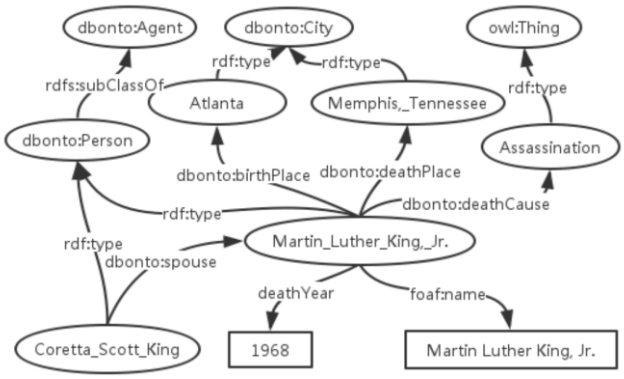


Figure 1. RDF snippet example

In examining the sample knowledge base, we find that the information in the KB is usually entity-centric, and many triples describe the property (attribute and relation) value pairs of an entity. In studying the sample question, we find that the question is always around an exact entity, and the supposed answer is always a property value of this entity. We consider the words that describe an entity in the question as an entity mention, and the words that describe the property in the question as the property mention. In the above sample question, "martin luther king" is supposed to be the entity mention, and "assassinated town" the property mention. We need to search the referent entity in KB to find the referent property of the entity, return the value of the referent property as answer, and ultimately answer the question.

The information included in Fig. 1 is sufficient to answer the question, and the right answer should be "Memphis Tennessee". However, arriving at the answer is difficult because a large gap exits between the question and KB. Three challenges must be overcome to solve the problem. The first challenge is to locate the entity mention and the property mention in the question. The second challenge is to search the referent entity in a large dataset. The third challenge is to answer the question; finding the referent properties from the large vocabulary in the KB to match the

978-1-5090-4174-9/16 $31.00 © 2016 IEEE
DOI 10.1109/ICCIA.2016.10

properties described in the utterance remains a difficult problem.

To address the first challenge, we depend on the natural language parser and predefined templates to obtain the entity and property mention. To address the second challenge, we collect the triples that describe only one entity together and build an entity-centric index to help search the referent entity, given that the entities are described by their attributes and their relationships using RDF triples. To address the third challenge, we expand the property mentions in the question and then match them to the properties of the referent entity. The matched property values are then returned as the answer.

The remainder of this article is organized as follows: Section 2 introduces the related work. Section 3 proposes the entity- centric index model. Section 4 presents the method to locate the entity mention in the question. Section 5 introduces the method to return the proper property values as answer, with the help of WordNet [1] and ConceptNet. Section 6 details the experimental results of our approach. Section 7 concludes the study.

II. RELATED WORK

Some researchers deal with semantic parsing methods for question answering. A question is mapped to its formal meaning representation and then translated to a KB query. The answers are retrieved by executing the query. These works focus on the semantic parsing methods to deal with the question. Some methods [2] use a domain independent meaning representation derived from the combinatory categorical grammar (CCG) parses. Method [3] develops semantic parsers for large databases based on a reduction to standard supervised training algorithms, schema matching, and pattern learning.

Some approaches [4]-[6] uses the knowledge base to help prune the search space when forming the parse. Work [7] researches how to narrow down the huge number of possible logical predicates for a given question. They build a coarse mapping from phrases to predicates using a knowledge base and a large text corpus. Then, they use a bridging operation to generate additional predicates based on neighboring predicates. Work [8] proposes a semantic parsing framework for question answering using a knowledge base. Semantic parsing is reduced to query graph generation, formulated as a staged search problem. They leverage the knowledge

III. ENTITY-CENTERED INDEX MODEL

In this section, we introduce the entity-centered index method for QA. We find that a target entity always exists in a question and that some of the property values of this target entity are supposed to be the answer. Based on this observation, we consider the entity to be the basic index unit.

For an entity, many property value pairs exit. A natural solution represents each entity using a fielded structure, where each field corresponds to property and field value corresponds to the property value. These representations can then be ranked using any fielded document retrieval model, such as BM25F. However, with this approach, the number of document fields soon becomes very large. To address this problem, we group the properties into multiple categories to reduce the fields' number and preserve some of the original structure.

We group RDF properties into five fields for a given entity *e*, and these fields and their values are listed as follows:

- *Type*: The *Type* field represents the *rdf:type* attribute of entity *e*. If the type of entity *e* is claimed as class *C*, then this field values are class *C* and its super class.
- *Name*: The *Name* field collects the name attributes of entity *e*, and we consider the name attribute as *foaf:name* or *rdfs:label* or the attribute ends with "name", "label", or "title". The field values are the literals parsed from the name attribute.
- *Attribute*: The *Attribute* field collects the literal attributes except for the name attributes. The field values are the corresponding attribute values, and the attribute names are also indexed for QA.
- *OutRelation*: The *OutRelation* field collects the object attributes from entity *e* to other entities e_i. The field values are a list of items, and each item represents object attribute information for entity *e*. The item is composed of the object attribute name, the name of entity e_i, and the class and super class type of entity e_i.
- *InRelation*: The *InRelation* field collects the object attributes from other entities e_i to entity *e*. The field values are a list of items, and each item is composed of the object attribute name, the name of entity e_i, and the class and super class type of entity e_i.

TABLE I. ENTITY-CENTERED INDEX MODEL

Fields	Value
Type	Person Agent
Name	Martin Luther King Jr
Attribute	deathYear:1968
OutRelation	birthPlace: Atlanta{City} deathPlace: Memphis, Tennessee{City} deathCause: Assassination{Thing}
InRelation	spouse: Coretta Scott King{Person, Agent}

Table I is the entity-centered index of Fig. 1. For the QA problem, the *type* and *name* field values are used to search the target entity, and the *attribute*, *outRelation*, and *inRelation* field values are used to search the target property and return the answer.

IV. LOCATING THE ENTITY MENTION IN THE QUESTION

We propose an entity mention locating approach, which identifies the entity mention in the question based on templates. We apply the Stanford Parser [9], which is a lexicalized dependency natural language parser, to parse the question. We then use a series of predefined templates to identify the entity mention.

Given a question "what town was martin luther king assassinated in?" We find that the entity mention should be "Martin Luther King" (a person) and that answer should be

the value of the property "assassinated town". Fig. 1 shows the dependency tree parsed by the Stanford Parser for the sample question. For the sake of simplification, we cut the leaf nodes with stop words, except for question words such as what, where, etc. After cutting the red dashed nodes, only two subtrees remain in Fig. 2. Except for the question subtree, which has a question word as the leaf node, the other subtree (i.e., the blue dashed rectangle) is the entity mention.

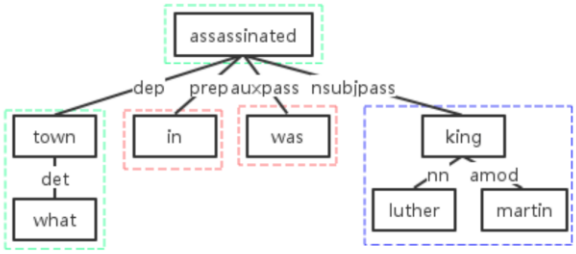


Figure 2. Example of dependency tree

In a dependency tree, we consider the node with a *subj/ obj/prep* edge from its father as a *subj/obj/prep* node. For example, node "king" is a *subj* node in Fig. 1.

We deal with questions with different question word in a different way. First, we cluster the questions using its question word. For each question word, we define a series of templates to locate the entity mention. Given that "WHAT" is the most useful question word, the entity mention locating templates are listed in Fig. 3.

Considering the different dependency tree roots, we define three templates to locate the entity mention.

- If the tree root is a be verb, the question usually looks like "What PROPERTY be *prep* ENTITY". Preferentially, the subtree with the *prep* node as root is defined as the target entity. If there is more than one *prep* node in a dependency tree, we select the highest-level *prep* node as the root, and its subtree is defined as the entity mention. Second, if no *prep* node exists in a dependency tree, the subtree with the highest level *subj* node as root is defined as the entity mention.
- If the tree root is the question word, the question usually looks like "What be PROPERTY *prep* ENTITY". The entity mention locating template is similar to the last template.
- If the tree root is a verb, the question usually looks like "What PROPERTY do ENTITY *verb*" or "What do/ be ENTITY *verb*". First, the subtree with the highest-level *subj* node as root is defined as the entity mention preferentially. Second, the subtree with the highest-level *obj* node as root is selected if no *subj* node exists. Finally, the subtree with the highest-level *prep* node as root is selected if no *subj* and *obj* nodes exist.

With these predefined templates, the entity mention can be located. Based on the index method proposed in Section 3, we index all the entities in the dataset by Lucene. We take the entity mention as the keyword to perform a search in Lucene. The type and name fields are used to match the search keyword in the target entity-locating stage. The entities are ranked according to the relevance score.

V. QA WITH PROPER ATTRIBUTE VALUE

After the target entity is returned, the next step is to extract the target attribute values to answer the question. Two challenges must be addressed here: One is to parse the at- tribute mention from a question, and the other is to match the attribute mention to the attributes of the target entity.

To identify the attribute mention from a question, we con- sider the entity mention locating templates. After the entity mention is located, we apply different attribute mention identifying methods when the dependency tree root is different.

- If the tree root is a be verb, we collect the nodes (except for the stop words) in the "what" branch as the attribute mention.
- If the tree root is a question word, we collect all the other nodes (except for entity nodes and the stop words) as the attribute mention.
- If the tree root is a verb, we collect the root node and the nodes (except for the stop words) in the "what" branch as the attribute mention.

After the attribute mention is parsed from a question, matching it to the attributes of a target entity remains a difficult problem. The attribute mention that occurs in a question may be very flexible in natural language. For example, the parsed attribute mention is "assassinated town" for the sample question in Fig. 2, but no such attribute exists for the target entity Martin Luther King, Jr. Observing the properties of the entity Martin Luther King, Jr. in Fig. 1, we find that the target property should be deathPlace.

To match the attribute mention to the attributes of a target entity, we expand the attribute mention by importing its synonyms and related words. WordNet is used to obtain the synonyms. A total 155287 unique strings and 117659 synsets exist in the newest WordNet version. Aside from WordNet, ConceptNet is used to obtain the related words and similar words for attribute mention. ConceptNet is built from nodes representing words or short phrases of natural language and the labeled relationships between them.

With the expansion of the attribute mention, we match the expansion to the target entity property in the *attribute*, *outRelation*, and *inRelation* fields. The matched attribute values of the target entity are returned as the question answer.

VI. EXPERIMENTAL STUDY

In this section, we introduce the dataset, evaluation metric, and evaluation results. We use Jena toolkit (jena.sourceforge.net) to manage RDF data. The experiments are developed within the Eclipse environment and on a 64 bit quad Core ThinkStation with 3.10 MHz and 16 GB RAM (14 GB of which are assigned to Java virtual machine).

We use the WEBQUESTIONS dataset [10], which consists of 5,810 question/answer pairs, as test questions. The questions are split into training and testing sets, which

contain 3,778 questions and 2,032 questions, respectively. The testing sets are selected as our questions.

The real-world RDF dataset DBpedia is selected as the KB to answer the questions. The DBpedia 2015-04 version de- scribes 4.8 millions instances in a total of 102 millions triples. Based on the testing question sets and the DBpedia knowledge base, we evaluate our target entity-locating method and QA method. The precision metric is used to evaluate the target entity-locating method. Given that the target entity is returned by Lucene, we evaluate the precision at different k (k is the number of entities returned by Lucene). The precision metric is also used to evaluate the QA method.

TABLE II. PRECISION OF TARGET ENTITY-LOCATING METHOD

Question word	Top-10	Top-20	Top-50	Top-100
what	58	62	68	72
who	58	61	64	67
where	57	61	66	69
total	58	61	66	69

Table II shows the precision of the target entity-locating method. We list the precision for three frequently asked question words. For every question word, we list the precision when the top-k entities are returned. In the target-entity locating stage, for a tested question, the result is considered to be right when the right target entity is returned in the top-k results. The precision refers to the result of the number of right questions divided by the total number of questions. Taking the question word "what" as example, the precision is 58% when the top-10 entities are returned by Lucene. This means that for a total of 1127 questions with the "what" question word, 58% questions receive the right returned entities in the top-10 results.

TABLE III. PRECISION OF THE QA METHOD

Question word	Top-10	Top-20	Top-50	Top-100
what	43	46	49	52
who	40	43	45	47
where	55	59	64	66
total	46	49	53	55

We find that the precision increases with an increase in the number of returned entities in Table II. Naturally, the precision is higher when more entities are returned. Even when only ten entities are returned, the precision reaches at 58% in total.

Table III shows the precision of the QA method. We also list the precision when the top-k entities are returned for three frequently asked question words. In the QA stage, for a tested question, the answer is considered to be right when the right target entity is returned in the top-k results and the right target value is selected. The precision is the result of the number of right answers divided by the total number of questions. We find that our QA method is promising when the precision of the target entity-locating method improves.

VII. CONCLUSIONS

In this study, we propose a QA approach on the DBpedia dataset. Based on the dependency parsing tree of the question and predefined templates, we obtain the entity and property mention from the question. We then stage the QA process into two steps: searching the target entity in KB and returning the target property values of the searched entity as the answer. In the first stage, we build an entity-centric index to help search the target entity. In the second stage, we expand the property mention with WordNet and ConceptNet to help match it to the searched entity property, and the matched property values are returned as the answer. The results show that our approach is promising in dealing with QA on Linked Data.

ACKNOWLEDGMENT

The work is supported by the Natural Science Foundation of Jiangsu Province under Grant BK20140643 and the National Natural Science Foundation of China under grant No. 61502095.

REFERENCES

[1] G. A. Miller, "WordNet: A Lexical Database for English," Communications of the ACM, vol. 38, no. 11, pp. 39-41.

[2] S. Reddy, M. Lapata, and M. Steedman, "Large-scale semantic parsing without question-answer pairs," Transactions of the Association for Computational Linguistics, 2014, pp.377-392.

[3] Q. Cai and A. Yates, "Large-scale Semantic Parsing Via Schema Matching and Lexicon Extension," Proc. of the 51st Annual Meeting of the Association for Computational Linguistics, 2013, pp. 423-433.

[4] H.Poon, "Grounded Unsupervised Semantic Parsing," Proc. of the Annual Meeting of the Association for Computational Linguistics, 2013, pp. 933-943.

[5] X. Yao and B. V. Durme, "Information Extraction over Structured Data: Question Answering with Freebase," Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014, pp.956-966.

[6] J. Bao, N. Duan, M. Zhou, and T. Zhao, "Knowledge-based Question Answering As Machine Translation," Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014, pp. 967-976.

[7] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic Parsing on Freebase from Question-Answer Pairs," Proc. of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp.1533-1544.

[8] W. Yih, M. Chang, X. He, and J. Gao, "Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base," Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language, 2015, pp. 1321-3331.

[9] D. Chen and C. D. Manning, "A Fast and Accurate Dependency Parser using Neural Networks," Proc. of the Conference on Empirical Methods in Natural Language Processing, 2014, pp. 740-750.

[10] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic Parsing on Freebase from Question-answer Pairs," Proc. of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp.1533-1544.

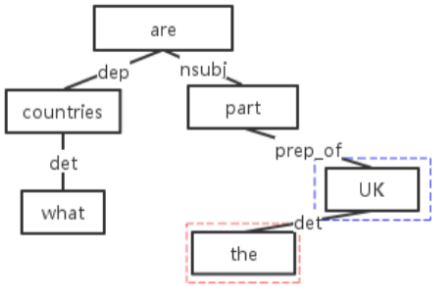
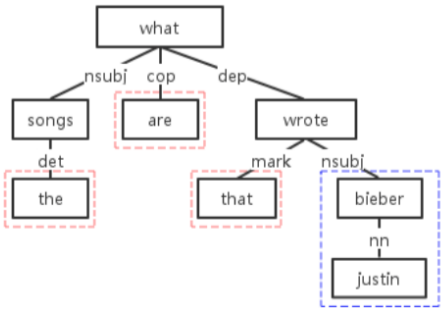
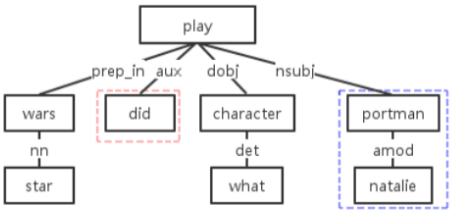
	Entity mention	Condition	Dependency tree	Example
1	The subtree with the highest-level *prep/subj* node as the root	Root is a be verb		What countries are part of the UK?
2	The subtree with the highest level *prep/subj* node as the root	Root is a question word		What are the songs that justin bieber wrote?
3	The subtree with the highest level *subj/obj/prep* node as the root	Root is a verb		What character did natalie portman play in star wars?

Figure 3: Entity mention locating templates for WHAT