# An Inference based Query Engine for RDF data

Mamdouh Farouk, Mitsuru Ishizuka
Creative Informatics Department
Graduate School of Information Science & Technology, The University of Tokyo
Tokyo, Japan
mamdouh@mi.ci.i.u-tokyo.ac.jp, ishizuka@i.u-tokyo.ac.jp

***Abstract*— Since the start of linked data project, more and more people publish their data in the linked data cloud as RDF data. Searching RDF data is important to exploit this massive amount of well-represented data. Finding implicit data is one challenge in searching RDF data. This work proposes an inference-based engine for SPARQL queries. Using inference our engine can answer some queries that cannot be answered by the normal SPARQL engines. The proposed engine expands the user query using the set predefined rules. The prototype for the proposed approach shows the effectiveness of adding inference to SPARQL engine.**

***Keywords*— SAPRQL query; search RDF; inference**

## I. INTRODUCTION

Maturity of semantic web languages helps the start of linked data evolution [1]. In the web of data, data represented as RDF triples <subject, predicate, object>. These triples are linked to each other constructing linked data cloud. The new web should be completely machine-understandable [2]. Since the start of Linking Open Data project, more and more providers publish linked data causing fast growing to linked data cloud.

On the other hand, one of the most important tools of Internet is web search. Almost all Internet users use web search to get their needs. The users want to write a query and find the exact answers. Consequently, improving searching RDF data is an urgent task. One challenge of searching web of data is getting implicit answers. In other words, search engines should go behind the raw data to understand web data and get query answers. Moreover, there is urgent need for a smart search techniques that make the use of the new evolution of linked data.

There are a lot of woke in searching RDF data. Many of these research based on SPARQL query language [3][4]. SPARQL is a query language for RDF. There is a similarity between SPARQL and SQL [1]. There are many SPARQL endpoints attached to linked data sets to facilitate querying RDF data sets. The user can submit SPARQL query to these endpoints and get the results via http.

The main objective of converting web data to RDF is to enable web agent to understand this data. However, web agents, that query linked data, cannot deeply understand RDF data. For example, consider a user queries the corpus of semantic web conferences, which contains information about some conferences in semantic web filed, to get information about authors who are interested in *semantic data representation*. Although, the corpus contains the needed information, the user may obtain no results. This is because a query engine cannot get the implicit answers.

Moreover, there are some issues should be considered in publishing and querying linked data. Selecting ontology is a difficult issue when someone wants to publish his data as Linked data. This is because there are different ontologies covering same vocabularies. Same situation when the user queries a dataset. The user has to know the ontology, which is used to represent the dataset, and he cannot use different equivalent vocabularies. Moreover, SPARQL endpoints return no result for the queries that uses different vocabulary, even though the dataset contains the answers. For example, the datasets in Dog Food server, http://data.semanticweb.org, uses swc:hasTopic property to represent the topics of a paper. However, if the user queries these data sets using dc:subject which is equivalent to swc:hasTopic, he will receive no result answer. In such case, the user is restricted to use same vocabulary as dataset. Furthermore, if the user wants to query different datasets with same query, the user has to write a different query for each datasets depending on the used vocabularies in each dataset. The search engine should allow some tolerance to query vocabularies especially because both human and machines are the users of this search service. Another important issue is that the search engine may return no result even the data contains the answer implicitly. This is because of the lack of inference.

This paper proposes adding inference to normal search engine to infer implicit data. Using inference enables search engine to answer some queries cannot be answered by normal engine. The proposed engine gets the implicit data based on a set of predefined rules. The proposed engine can query RDF data set or SPARQL endpoint. It is developed based on Jena SPARQL API with additional inference step. Using the implemented inference step the engine expand user query to a more detailed queries that can be answered with the normal engines.

SPIN is a set of ontology properties that enable user to attach rules to ontology [5]. Using these rules infers more knowledge or generates data dynamically. However, our approach separates between ontology level and rule level. Therefore, our approach avoids rules conflicts in ontology level.

978-1-4673-1090-1/12/$31.00 ©2012 IEEE

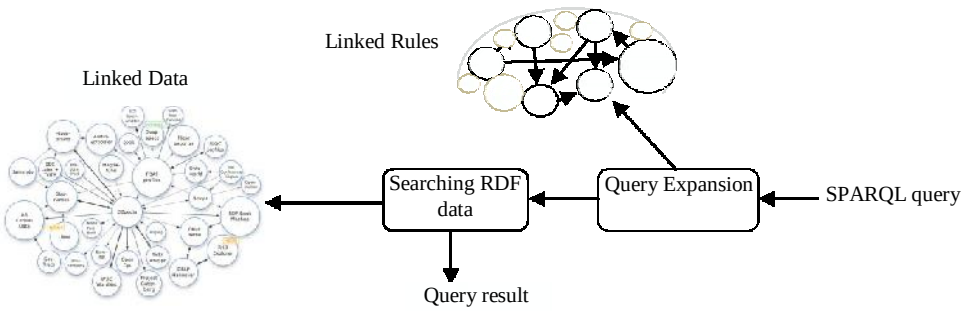


Figure 1. System architecture

The reminder of this paper is organized as follows: section 2 shows a general overview to the proposed system. Section 3 explains the idea of adding user-defined rules. The proposed search engine is described in section 4. Section 5 discusses the results of the proposed approach. Section 4 concludes the work

## II. System Architecture

Improving web understandability is an important step after maturity of linked data. Web agents should reason over RDF data to go behind the raw data and infer the implicit information. Based on better understanding to RDF data web agents will behave smartly. In order to improve understandability of RDF data, two main things are required. These things are inference engines and reasonable amount of knowledge (rules). This paper focuses on inference on RDF data based on user defined rules.

Inference on semantic web is vital to enable agents to deeply understand the web of data. Moreover, inference on semantic web is characteristic by discovering new relationships between web resources. Using inference in linked data search improves query result and enables search to get the implicit answers.

The proposed system is a search engine for RDF data. The proposed search engine exploits the user-defined rules to get better answers for user queries. However, backward chaining is used to expand SPARQL query.

The proposed system consists of two main parts as shown in figure 1. The first component is normal query execution. The second component in which our contribution is applied is query expansion based on backward chaining of a predefined set of rules. The query expansion outputs a list of new queries. The proposed engine recursively run the new queries and accumulates the results of all these queries.

## III. Adding Rules

Data publisher may add extra knowledge (rules), which considered as an extension for the original data. Moreover, adding this kind of enables the data admin, who selects specific vocabularies to represent the published data into RDF, to suggest alternatives vocabularies and expresses them in rules instead of duplicate RDF data with different vocabularies. The SPARQL endpoint admin should add a set of rules depending on the meaning of RDF dataset and common queries that users used to ask.

Moreover, there are two types of rules that can be added to the proposed wrapper. The first type focuses on discovering new RDF triples to get the implicit data, for example, the rules that infers the relation between topics and authors. This rules defined as: *if a person X is an author for the paper Y and the topic of the paper Y is T then the person X is interested in the topic T*. The other type of rules focuses on vocabularies mapping to allow client to use some different equivalent vocabularies. This will not restrict the user to use the same vocabulary as RDF dataset.

The format of user-defined rule is a simple production rule, "condition → action", designed to be easy for reasoning. The condition part syntax is the same as SPARQL query condition syntax. The action part is also represented into SPARQL syntax. Figure 2 shows an example for the rule: If a person A is an author to a paper Y, and a person B is an author to the paper Y, then → A knows B. The first part of the rule format is xml namespaces for the used vocabularies. The second part

```
<rule id="2" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
   xmlns:foaf="http://xmlns.com/foaf/0.1/"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:iswc="http://annotation.semanticweb.org/iswc/iswc.daml#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
   <text>
    if two people are authors for the same paper, then they know
each others.
   </text>
<condition>
<con>
{
 ?ppr rdf:type iswc:InProceedings.
 ?person rdf:type foaf:Person.
 ?person2 rdf:type foaf:Person.
 ?ppr dc:Creator ?person.
 ?ppr dc:Creator ?person2.
 FILTER (?person != ?person2)
}
</con>
</condition>
<action>
{   ?person foaf:knows ?person2.  }
</action>
 </rule>
```

Figure 2. Example of user-defined rules

is the condition of the rule. The last part is the action part, which should be true if the conditions are true.

Using SPARQL-syntax query contributes solving the problem of user restriction in using ontologies. The user can use equivalent ontology properties to query RDF datasets. For example, if a rule is added to the RDF dataset which gives some alternative to the property swc:hasTopic such as foaf:topic, or dc:subject, this will give the query party some flexibility to ask with any of these properties.

Moreover, using these rules during the processing of original data to infer more data on the fly costs overhead processing time. However, using inference rules during searching give better results. In addition, the proposed approach avoids data duplication and keeps the data size. The size of data should be in a reasonable range that does not affect the web agent performance [6].

## IV. SPARQL ENGINE+

The proposed engine is an extension to the normal SPAQRQL engine. An inference step is added to the SPARQL engine to make the use of the user-defined rules and answer some queries that cannot be answered using normal engines. Without a simple inference step, the engine cannot answer some query that can be answer based on the available datasets. Sometime the dataset contains the desired result but the engine cannot extract it. Using inference, the engine goes behind the raw data to find query answer.

The proposed engine can query RDF data set or SPARQL endpoint. It is developed based on Jena SPARQL API [10] with additional inference step. The actual usage of the added inference step is not to infer more data. However, the inference step is used for query expansion process to get more detailed queries that can be answered using the normal engines.

The user can custom the engine behavior though options panel. The user determines when the engine should apply inference and to what extent. For example, the engine can be adapted to run inference only in case of no result returned by normal search. In addition, the user may stop the inference whenever a result comes. The default is that the engine will get all possible solutions.

There are some advantages of using the selected rule syntax. It is easy for user to write his rules with no need to learn a new syntax. SPARQL syntax rules are very convenient to do data driven on RDF datasets. In addition, SPARQL syntax rules are much easier to make SPARQL query expansion.

### *A. Backward chaining*

The proposed approach applies backward chaining in query answering process for query expansion. The algorithm of SPARQL query expansion is a recursive algorithm that gets all possible queries based on a set of selected rules a set of predefined rules. Indexing for the published rules are established to link different rules based on the inferred relations. This index for the published rules facilitates finding the appropriate rules to expand a SPARQL query. The basic

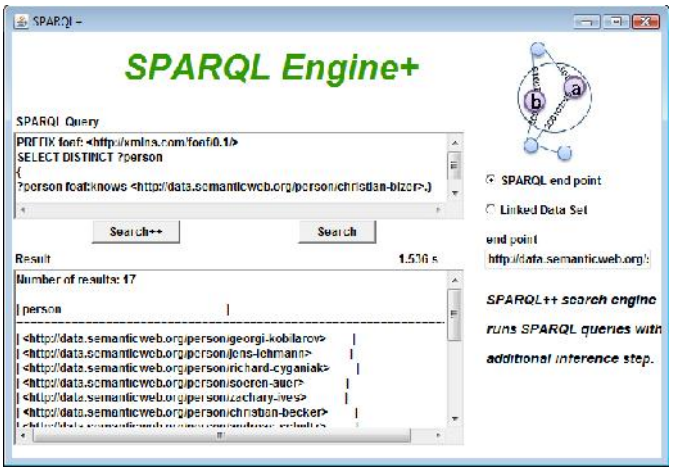


Figure 3 proposed SPARQL engine interface

idea of this query expansion is to replace query condition with other conditions based on backward chaining of the rules.

Query expansion based on backward chining algorithm is as follows:
Input: SPARQL query, rules in shared rule cloud
Output: list of new queries equivalent to the inputted query

- Get list of properties used in query conditions.
- Get related rules that can be used to expand the inputted query
- For each rule in the related rules list
  - Match between rule actions and query conditions
  - Bind matched variables and keep in a mapping state
  - Replace the matched query conditions with rule premises
  - Recursive call to expand the new query // this call start matching the new query and only the rest of rule set
- Combine all new queries in one list

The matching process in this algorithm depends on between corresponding component in the triples. This means subject with subject and so on. After matching a rule with a query, new conditions should be constructed for the new query. In these conditions, matched variables should be substitute to be like the ones that are used in the original query. Also the prefixes of the newly added condition should be included in the constructed query.

A mapping state holds the mapping between different matched objects in order to construct a proper query that gets the answer of the user query.

## V. EXPERIMENT

The proposed SPARQL engine is implemented using Java, figure 3. In order to show the effectiveness of the proposed approach, the developed prototype is tested on real data sets. It is used to query semantic web conferences corpus. This corpus

exits in Dog food server, www.data.semanticweb.org, which contains a large RDF datasets. It contains 180445 unique triples. It provides SPARQL endpoint, which is used to query the contained datasets. These datasets contain information about some conferences related to semantic web field. They contain data about 3000 paper, 7500 people, and 2100 organization at 28 conferences and 148 workshops.

In this experiment, the developed wrapper uses a set of five rules. Example of these rules is:

- If a person A is an author to a paper Y, and a person B is an author to the same paper Y → A knows B.
- If a person A is an author to a paper Y, and the main topic of Y is T then → A is interested in T.
- If a topic X is a keyword for a paper Y → topic X is a subject of Y.

The first two rules infer implicit data. However, the third rule maps ontology vocabularies to facilitate finding query answer.

Table 1. List of Queries used in the experiment

| Number | Query |
|---|---|
| Q1 | Who knows Prof. Chris Bizer |
| Q2 | Who is interested in Semantic web |
| Q3 | Get all papers in Application software |
| Q4 | Who is interested in and Application software and knows Prof. Bizer |
| Q5 | Who is interested in semantic web and speaks French language |
| Q6 | Get all papers in ISWC 2010 |
| Q7 | Who is interested in semantic web and speaks French and knows Prof. Anthony Ventresque |

During this experiment, we run seven different queries using our approach and using normal SPARQL engine. These queries were selected to test different cases. Table 1 shows the list of queries used in this experiment. Results are shown in table 2.

The first query in the above list asks about the people who know prof. Charis Bizer. The proposed engine returns 17 results for this SPARQL query. However, the normal engine does not return any result.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?person
{?person foaf:knows
<http://data.semanticweb.org/person/christian-bizer>.}
```

Using the user-defined rules in SPARQL engine gets better result and improves the query answer process. Moreover, the proposed engine can answer some queries that cannot be answered by normal search engines. Execution time almost the same in case of no matched rules. Even though we did not pay much care about improving performance of the developed engine, the overhead in execution time in case of rule matching is still in the acceptable range.

Table 2. Query result using normal engine and the proposed approach

| Query number | Normal search | | Proposed search | | |
|---|---|---|---|---|---|
| | Number of results | Execution time | # of generated queries | Total number of results | Execution time |
| Q1 | 0 | 0.54 | 1 | 17 | 1.299 |
| Q2 | 0 | 0.56 | 1 | 831 | 3.342 |
| Q3 | 17 | 0.843 | 0 | 17 | 0.855 |
| Q4 | 0 | 0.55 | 3 | 9 | 16.908 |
| Q5 | 0 | 0.56 | 5 | 17 | 5.028 |
| Q6 | 87 | 1.092 | 0 | 87 | 1.103 |
| Q7 | 0 | 0.56 | 11 | 8 | 8.133 |

## VI. CONCLUSION

One of the main objectives of using RDF to represent web data is to enable machines to understand web data. However, searching RDF is lake of an inference to understand RDF data deeply. This paper proposes using inference in SPARQL query answering to enables search engine to find more answers and improve the recall of searching technique. Using inference facilitates understanding web data. The proposed engine expands SPARQL queries based on backward chaining of a set of predefined rules. The developed prototype shows the effectiveness of using inference during searching RDF data.

## REFERENCES


[1] Axel Polleres, From SPARQL to rules (and back), Proceedings of the 16th international conference on World Wide Web (WWW2007),, May 2007, Banff, Canada pages 787–796

[2] Arup Sarkar --- Ujjal Marjit --- Utpal Biswas “linked data generation for the university data from legacy database” International Journal of Web & Semantic Technology Year: 2011 Vol: 2 Issue: 3 Pages/record No.: 21-31

[3] S. Elbassuoni, M. Ramanath, R. Schenkel, and G. Weikum. Searching rdf graphs with SPARQL and keywords. IEEE Data Engineering Bulletin, 33(1), 2010

[4] Olaf Hartig, Christian Bizer, and Johann-Christoph Freytag: Executing SPARQL Queries over the Web of Linked Data. In Proceedings of the 8th International Semantic Web Conference (ISWC), Washington, DC, USA, Oct. 2009

[5] 9. Holger Knublauch, James A. Hendler, Kingsley Idehen “SPIN - Overview and Motivation”, http://www.w3.org/Submission/2011/SUBM-spin-overview-20110222/ , February 2011

[6] J. Minsu and J.C Sohn, "Bossam: An Extended Rule Engine for OWL Inferencing," In Proc. RuleML 2004, pp. 128-138, 2004.