2015 17th UKSIM-AMSS International Conference on Modelling and Simulation

Result Optimisation for Federated SPARQL Queries

Arooj Fatima, Cristina Luca, George Wilson, Mohamed Kettouch
Department of Computer Science
Anglia Ruskin University
Cambridge, UK
arooj.fatima@anglia.ac.uk, cristina.luca@anglia.ac.uk, george.wilson@anglia.ac.uk,
mohamed.kettouch@student.anglia.ac.uk

Abstract— **Linked open data end-points are accessible to users and applications to exploit semantic data. However it is not very usable for non-expert users. A search tool can be evaluated on certain criteria i.e. access to knowledge resources, usability, expressivity and scalability. For a user friendly semantic search tool, there is a number of challenges e.g. translating user query to formal SPARQL query, scalability, guidance, optimising query results for better readability and visualisation etc. The focus of this paper is the result optimisation for the data received as a result of running queries over different SPARQL end-points. The paper highlights two aspects of the result optimisation i.e. (i) result ranking and (ii) result readability. The paper introduces algorithm for result ranking and suggests the layout schema that defines the result template for the user interface.**

Keywords- Semantic web; Result optimisation; result ranking; ontologies

I. Introduction

SPARQL has become the standard language for querying RDF-based semantic data. Since 2008, the features of SPARQL have been the focus of interest for the semantic research communities. As a result of these efforts, the theoretical properties of the SPARQL are now well-understood [11]. Currently, SPARQL is the core technology for querying the semantic web and most of the RDF-based applications are accessible via SPARQL endpoints. However the Linked Open Data is still only usable for semantic technology experts and programmers. The problem of friendly user interfaces for accessing Linked Data is still largely unsolved [21].

One important feature for the user friendly interfaces is 'Result ranking'. There is a number of search tools like Swoogle [18], Sindice [19] and Watson [20] that now include large repositories of ontology data sets and the size of data is increasing constantly. This rapid increase of semantic data adds a need for procedures to rank results and display the results in a user friendly way.

This work focuses on two aspects of result optimisation: (i) result ranking and (ii) schema for result display layout. This paper is divided as follows - Section II and III describe the challenges faced by a semantic search tool focusing more on the end user interface and the basic definitions used in the paper. Section IV analyses the related work done in the similar context followed by Section V that explains the proposed system. Finally, Section VI introduces an algorithms along with some examples for result optimisation.

II. Challenges

There is a number of challenges faced by the end user of a semantic search tool:

i) Query translation – a facility to translate from informal end user query to formal SPARQL query. From the end user point of view, the semantic data is difficult to access given that a non-expert user may not have the required technical knowledge to express their needs in a relatively complex query language, such as SPARQL. This is why translating user queries is one of the biggest challenges for a semantic search tool.

ii) Scalability – a measure of a system to cope with large datasets. Scalability is a challenge for linked open data repositories with the increasing number of data sets.

iii) Guidance – to help end users to build proper queries using search features i.e. auto-complete and auto-suggest. The implementation of such functions requires the knowledge of the datasets and their specified structure [2].

iv) Result ranking – in a traditional search environment, the results are ranked based on the relevancy, popularity and customisation factors. For a semantic search tool however, result ranking is a challenging task as whilst the semantic search results are assumed to be relevant by default, there is a need to do some further verifications to confirm the relevancy of results.

v) Result readability – the results returned from a SPARQL endpoint are commonly in JSON, XML or HTML format. For a better visualisation and readability, the results need to be re-formatted.

In the present work, the authors aim to resolve the problem of result optimization and propose a solution that aims to optimize results fetched from different SPARQL end-points. The main focus of this work is the features of result ranking and result readability.

978-1-4799-8713-9/15 $31.00 © 2015 IEEE
DOI 10.1109/UKSim.2015.50

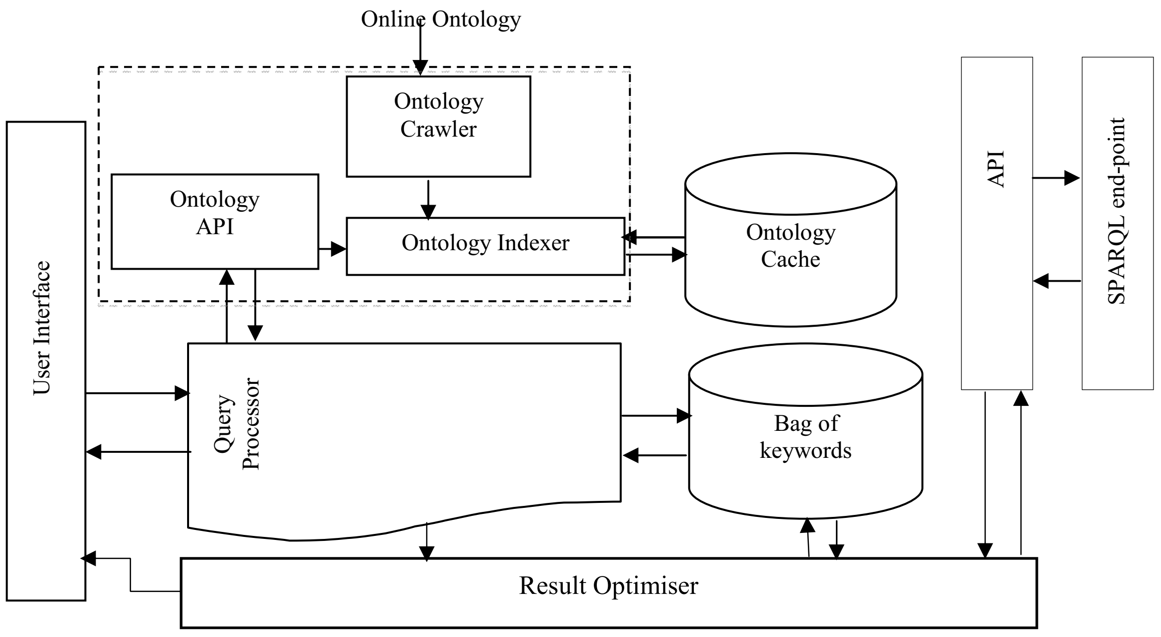

Fig. 1. Framework for query processing and result ranking

III. Preliminaries

A. Ontologies

Semantic Web is designed to build a web of meaning. The foundation of effective communication on the Semantic Web is ontology, sometimes defined as a vocabulary of data. Ontology provides a formal, explicit specification of a shared conceptualisation of a domain [3]. An ontology facilitates knowledge sharing over distributed systems; in other words, it allows systems or applications to communicate even when these are not formerly designed to interoperate. It helps to create a common understanding of data among different applications.

B. SPARQL endpoints

The most common way for accessing RDF datasets is by means of SPARQL endpoints. The SPARQL endpoints are similar to APIs (Application Program Interfaces) that implement the SPARQL protocol and are accessible by other applications and users to query RDF data. Although there are often some limitations for some endpoints that restrict data access to a certain limit or set a maximum number of query execution in a specified duration [1], the endpoints are the common mean of accessing open linked data.

C. Federated query

A federated query is an extension of SPARQL query for executing queries distributed over different SPARQL endpoints. SPARQL 1.1 supports federated queries that merge data distributed across the Web using the 'SERVICE' keyword [5].

IV. related Work

Swoogle is a semantic web search engine that discovers and indexes information in semantic web documents. Swoogle provides semantic web data access service, which helps human users and software systems to find relevant documents, terms and triples [23]. However, Swoogle has a number of limitations i.e. limited quality control (redundant ontologies), limited search (keywords based only) and no support for a formal query language like SPARQL [22]. Sindice[19] is another example of popular semantic search engines, but that is not very usable for people with no understanding of semantic technologies.

Watson [20] uses keyword search feature that makes it similar in use with usual web search systems. Watson search tool matches set of keywords entered by the user to the literals of entities saved in semantic document and display entities matching each keyword. The search tool also provides facility to find data considering types of data i.e. classes, properties, individuals. However, it is not easy to discover data and results are not readable.

A number of efforts have been made towards the enhancement of page ranking techniques. Consequently, numerous algorithms have been proposed for page ranking [12]. One instance is [13] that focuses on the granularity of Web pages and use three kinds of information to re-rank documents i.e. document information, query Information and ancillary Information. The authors of [14, 15] propose inter-document relationships that can be obtained from the text, hyperlinks and other information and assign similarity weights to the documents. However these approaches are limited for a semantic result ranking. In paper [17] a ranking approach based on syntactic classification, still; this approach does not take into consideration the semantics of the data.

The idea proposed by [16] is partially similar to the authors' present work but they approach the problem differently. The solution proposed by [16] is to analyse on how the ontologies have been rated and reviewed by the users. It allocate weights for concept hierarchy using users' profile.

The above work originated from the similar idea to formalize semantic data to be made available online, for users and applications to find and exploit. However, they mostly rely on extracting labels or comments from RDF documents and match the search keywords with saved data. To enable applications to explore large scale semantic data, new mechanisms are required for query optimization and result optimisation.

V. Proposed System

In the present work, the authors proposed a ranking technique as a part of the result optimization system that is based on semantic classification. The authors propose an algorithm to rank ontology concepts that help to arrange results in an order of top ranked ontology. The authors also define result schema for a user friendly layout for the results.

The proposed solution is based on authors' previously modelled framework [9, 10]. The framework consists of a number of modules i.e. user interface, ontology processor, query processor, bag of keywords, API and result optimiser. The query processor generates a SPARQL query from the keywords entered from the user interface and sends the mapped query to the API. The API interacts with the SPARQL endpoint/s to fetch results matched to the user query. The results received are then sent to the result optimiser that formats the results and sends to the user interface.

This paper focuses specifically on three parts of the framework that are involved in result optimisation. These parts are:

A. *User Interface*

The user interface retrieves a query or search keyword/s from the end user and dispatches the keywords to the query processor. Once processed by other parts of the framework ('bag of keywords', ontology processor and result optimizer) the user interface displays the results based on a set schema (see Section VI (B) for details).

B. *Query Processor*

The query processor is the core of the entire framework. It performs a number of key tasks but this paper will focus only on the tasks that are involved in result optimisation. The first step is handled by the query processor which generates a SPARQL query – that is, a query statement is written that will return results ranked by relevance.

C. *Result Optimiser*

The result optimiser works in two ways. Firstly it interacts with the query processor to get schema attributes (defined in Table III), and secondly it facilitates result ranking and fetches schema from the 'bag of keywords' to set up the result layout.

D. *Bag of keywords*

'Bag of keywords' is one of the root concepts used in the prototype model. It is a repository of keywords with defined meanings attached using particular properties and classes.

VI. Result Optimisation

This section describes the prototype implementation based on the proposed model. Section (A) explains the ontology ranking technique followed by Section (B) that demonstrates the result schema. Finally section (C) introduces algorithms to rank ontologies and generate rank based SPARQL queries.

A. *Result Ranking*

Since semantic web data is based on ontologies, the ranking of the search results must be somehow ordered on the basis of semantic relevancy. Due to different interests and points of view, many people define their own ontologies for the same domain of interest e.g. the concept 'Person' has been defined in various namespaces i.e.

- http://xmlns.com/foaf/0.1/Person
- http://schema.org/Person
- http://www.w3.org/ns/person#Person.

The flexibility of defining different ontologies for the same concepts raises a challenge of ontology alignment and trust levels (i.e. which ontologies are the most trusted ones). Whilst the ontology alignment is not the scope of this paper, it focuses on the result ranking to get a level of trust by ranking popularity based results on the top. The proposed system does the ontological ranking that helps the result ranking. The proposed model saves records for the various aligned ontologies and their concept popularity. The concept popularity is measured by the number of usage on different domains. Table I shows some examples of the concept 'Person' ranked by the proposed system. It can be seen that http://xmlns.com/foaf/0.1/Person has the top rank because it is the most used namespace on different domains for this concept.

Similarly, Table-II displays an example of property ranking. Whilst the same property can be used with multiple classes in different namespaces, it is required to set some priority ranking for each combination of class and the property. From the given example (Table-II), the property http://xmlns.com/foaf/0.1/name has been used with multiple classes i.e. 'Person', 'Organisation', 'Group', 'Document' etc. The proposed system saves the counters for each set of property and class and ranks the highest counter on top.

The property ranking helps the search tool to rank results especially when there are no concepts identified from a user query e.g. for a search query 'Milli', there are no specified classes or concepts and the keyword ('Milli') is mapped to the property foaf:name (http://xmlns.com/foaf/0.1/name). The property foaf:name can be linked to multiple classes i.e. name of a 'Person', name of an 'Organisation' etc. In such a case the search system will find results in an order based on defined ranks. For the above example the proposed system will display names of the 'Persons' matched with the keyword on the top followed by the names of the organisations and so on.

B. Result schema

The SPARQL endpoint returns results in JSON, XML and HTML format and mostly it contains URIs for the resources. Since the proposed system is intended to be end-user focused the results need to be readable and visually user friendly. To achieve this the authors introduce a result schema. The schema defines key attributes for each class in an ontology with a pre-defined layout (as shown in Table III).

The query processor uses schema defined for a class while generating SPARQL query e.g. for the class 'Person' the schema attributes defined are name, givenName and familyName and the statements generated are

```
SELECT ?person ?name ?givenName ?familyName
?person a foaf:Person
?person foaf:name ?name
?person foaf:givenName ?givenName
?person foaf:familyName ?familyName
```

C. Algorithms

In this section the authors present the Algorithm 1.1 that describes the process of setting up ontology based ranking and the Algorithm 1.2 that generates query statements to fetch ranked results from SPARQL endpoints.

Algorithm 1.1 Ontology parser/ranker

```
Require: ontology, domain
1.  concepts_{0..n} = parse(ontology)
2.  for each concepts c   do
3.        get  saved_concepts
4.        if c exists in saved_concepts  then
5.             if  domain not listed then
6.                  add domain
7.                  concept_counter++
8.             else
9.                    // ignore, do nothing
10.            end if
11.       else
12.            if c mapped to saved_concepts then
13.                 add to aligned group
14.            end if
15.            list domain
16.            concept_counter++
17.       end if
18. end for
```

Algorithm 1.1 crawls (the for loop on line 2) through online ontologies used at a domain level and checks if the concept has already been recorded in 'ontology cache'. If concepts exists (line 4) and domain is already not listed, it increments the usage counter for that ontology concept. Greater is the number of usage counter, higher will be the concept ranking. If concept does not exists in ontology cache, the algorithm tries to align it with the existing concepts e.g. http://xmlns.com/foaf/0.1/Person is the same as http://schema.org/Person that is why both concepts are aligned and put in a group. The algorithm lists non-existing domain and initiates the usage counter with +1.

Algorithm 1.2 Rank based query generation

```
Require: sentence
Output: statements = null
1.  words_{0..n} ⟵ split(sentence)
2.  classes_{0..n} ⟵ getClasses(words)
3.  concept ⟵ findMainConcept(classes_{0..n})
4.  namespaces_{0..n} ⟵ getNameSpaces(concept)
5.   for each namespaces ns  do
6.        domains_{0..n} ⟵ getDomains(ns)
7.        create statements
8.  end for
9.  join statements
10. statements ⟵ getUniqueStatements()
11. return statements
```

The Algorithm 1.2 generates query statements to fetch ranked results from SPARQL endpoints.

The algorithm takes a search query received from the user interface. It splits the user query into search terms and finds the concepts (classes and properties) defined. The next step is to find the one main concept from the user query if more than one concept found. After finding the main concept, the algorithm fetches all the namespaces that define the same concept from the ontology cache e.g. Table-I displays the namespaces and their ranking for the concept 'Person'. The algorithm gets domains' URLs for each namespace. The ranking for the domains is calculated by the number of instances the domain got for the same namespace e.g. for the concept 'http://xmlns.com/foaf/0.1/Person' there are a number of listed domains as shown in Table IV

TABLE I. DOMAIN RANKING

Domain	**Instance count**	**SPARQL endpoint**	**Rank**
Freebase	3,401,174 [7]	http://www.freebase.com/base/sparql	1
DBPedia	1,450,000 [6]	http://dbpedia.org/snorql	2
British Library	1,211,026 (source: SPARQL endpoint)	http://bnb.data.bl.uk/sparql	3
…	…	…	…

TABLE II. CONCEPT RANKING

Concept URI	Rank
http://xmlns.com/foaf/0.1/Person	1
http://schema.org/Person	2
http://www.w3.org/ns/person#Person	3
http://dbpedia.org/ontology/Person	4
http://www.bbc.co.uk/ontologies/coreconcepts/Person	5
…	…

TABLE III. PROPERTY RANKING

Property URI	Class	Rank
http://xmlns.com/foaf/0.1/name	foaf:Person	1
http://xmlns.com/foaf/0.1/name	foaf:Organization	2
http://xmlns.com/foaf/0.1/name	foaf:Group	3
http://xmlns.com/foaf/0.1/name	foaf:Document	4
…	…	…

TABLE IV. RESULT SCHEMA

Class	Attributes	Schema layout
http://xmlns.com/foaf/0.1/Person	name, givenName, familyName	name(family Name, givenName)
http://purl.org/ontology/bibo/Book	title	title
…	…	…

Finally, the algorithm generates statements for each domain using 'SERVICE' keyword and the schema statements as follows

```
SERVICE < service URL> {
    // schema attributes for the class
}
```

For example the federated query generated for the class 'foaf:Person' using schema attributes and namespaces will be as follows

```
    PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
    PREFIX foaf:<http://xmlns.com/foaf/0.1/>

    SELECT ?person ?name ?givenName ?familyName
    WHERE {
          ?person a foaf:Person.
    SERVICE <http://www.freebase.com/base/sparql> {
        ?person foaf:name ?name .
        ?person foaf:givenName ?givenName .
        ?person foaf:familyName ?familyName
    }
    SERVICE <http://dbpedia.org/snorql/> {
        ?person foaf:name ?name .
        ?person foaf:givenName ?givenName .
        ?person foaf:familyName ?familyName
    }
    SERVICE <http://bnb.data.bl.uk/sparql> {
        ?person foaf:name ?name .
        ?person foaf:givenName ?givenName .
        ?person foaf:familyName ?familyName
    }
    }
```

VII. Possible Implementations

The proposed framework can be used as a back bone of the search tool for any system using semantic web technologies i.e. linked open data (LOD) and a SPARQL endpoint to access LOD data. The proposed framework contains a number of modules that may work independently with little configuration i.e. ontology processor, query optimizer, query formatter, API etc. This flexibility enables the framework to be used as a whole or as a part of other search tools.

The algorithm for the ontology ranking can be implemented in multiple ways to fit in a semantic search system.

VIII. Conclusion

Among various challenges for creating an efficient semantic search tool is the question of how to optimise query results (result optimisation). Result optimisation in a view of user friendliness involves result ranking and result layout for better visualisation of those results. SPARQL endpoints are the access points for Linked Open Data. Various Linked Open Data sets use various namespaces for similar concepts that raises the challenge of ontology alignment and result ranking. This paper has focused on those aspects of result optimisation that will give a more user friendly experience. The solution is based on a previously modelled framework that groups results in an order of popular ontological concepts, and rank domains for an ontology concept based on the number of instances found. This work has also described a layout schema for the captured results from the SPARQL endpoints.

References

[1] C. Buil-Aranda, A. Polleres and J. Umbrich, Strategies for Executing Federated Queries in SPARQL1.1. In proceeding of ISWC, 2014.

[2] S. Yamamoto, Human Interface and the Management of Information. Information and Knowledge Design and Evaluation: 16th International Conference, HCI International, Heraklion, Crete, Greece, June 22-27, 2014.

[3] R. Studer, R. Benjamins, and D. Fensel, 1998. Knowledge engineering: Principles and methods. Data & Knowledge Engineering, 25(1–2):161–198.

[4] W3C, SPARQL 1.1 Federated Query [Accessed Febraury 1st, 2015]. Internet, 2013.

[5] DBPedia, The DBPedia Ontology (2014) [Accessed Febraury 15, 2015]. Internet, 2014.

[6] Freebase, 2014. Commons domain [Accessed Febraury 12, 2015]. Internet, 2014..

[7] P. Hyunjung, P. Jinsoo, and S. Rho, A link-based ranking algorithm for Semantic Web resources: a class-oriented approach independent of link direction. Journal of Database Management, vol 22 issue 1, pp 1-25, 2011.

[8] G. Acampora, V. Loia, and A. Vitiello, Enhancing ontology alignment through a memetic aggregation of similarity measures. Information Sciences, [e-journal] 250 (0), pp.1-20, 2013.

[9] A. Fatima, C. Luca, C. and G. Wilson. User Experience and Efficiency for Semantic Search Engine (accepted). 14th International Conference on Optimization of Electrical and Electronic Equipment (OPTIM 2014). May 2014.

[10] A. Fatima, C. Luca, and G. Wilson, A New Framework for a semantic search Engine, In: Proceeding of 16th International Conference UKSIM, Cambridge, 2014 pp. 446-451.

[11] V. Egor and B. C. G. Kostylev, On the Semantics of SPARQL Queries with Optional Matching under Entailment Regimes. In the proceedings of 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014.

[12] J. Agarwal, N. Sharma, P. Kumar, V. Parshav, and R. H. Goudar, Ranking of Searched Documents Using Semantic Technology. Procedia Engineering, 64, pp. 1-7, 2013.

[13] D. Zheng, Research on Cross-Language Information Retrieval Based on a Combination of Ontology with Statistical Language Model. Dissertation for the Doctoral Degree in Engineering, Harbin Institute of Technology, pp 1-3 2006.

[14] J. Balinski and C. Danilowicz. “Re-ranking method based on inter-document distances”. Information Processing and Management, 41(2005), pages 759-775, 2005.

[15] V. Jain and M. Varma. Learning to re-rank: query-dependent image re-ranking using click data. In Proceedings of the 20th international conference on World Wide Web, 2011.

[16] C. Patel, K. Supekar, Y. Lee, and E. Park. Ontokhoj: A semantic web portal for ontology searching, ranking, and classification. In Proc. 5th ACM Int. Workshop on Web Information and Data Management, pages 58–61, New Orleans, Louisiana, USA, 2003.

[17] D. Mukhopadhyay, P. Biswas, Y. ChonKim, A Syntactic Classification based Web Page Ranking Algorithm, 6th International Workshop MSPT, 2006.

[18] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V.C. Doshi, J. Sachs, Swoogle: A Search and Metadata Engine for the Semantic Web. In: 13th ACM Conference on Information and Knowledge Management, Washington D.C. 2004.

[19] G. Tummarello, R. Delbru, E. Oren, Sindice.com: Weaving the open linked data. The Semantic Web, pp:552-565. Springer Berlin Heidelberg, 2007.

[20] M. d'Aquin, M. Sabou, E. Motta, S. Angeletou, L. Gridinoc, V. Lopez and F. Zablith, “What can be done with the Semantic Web? An Overview of Watson-based Applications,” 5th Workshop on Semantic Web Applications and Perspectives, SWAP Rome, Italy, 2008.

[21] P. Hoefler, Linked Data Interfaces for Non-expert Users. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC vol. 7882, pp. 702–706, 2013.

[22] E. Motta, M. Sabou, Next Generation Semantic Web Applications. In: Mizoguchi, R., Shi, Z.-Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185, pp. 24–29, 2006.

[23] T.W. Finin, L. Ding, R, Pan, A. Joshi, P. Kolari, A. Java, and Y. Peng, Swoogle: searching for knowledge on the semantic web, Proceedings of the National Conference on Artificial Intelligence (AAAI), pp.1682–1683, 2005.