LOQUS: Linked Open Data SPARQL Querying System

Prateek Jain
*Wright State University - Main Campus*

Kunal Verma
*Wright State University - Main Campus*

Peter Z. Yeh

Pascal Hitzler
pascal.hitzler@wright.edu

Amit P. Sheth
*Wright State University - Main Campus*, amit@sc.edu

Follow this and additional works at: https://corescholar.libraries.wright.edu/knoesis

Part of the Bioinformatics Commons, Communication Technology and New Media Commons, Databases and Information Systems Commons, OS and Networks Commons, and the Science and Technology Studies Commons

**Repository Citation**

This Report is brought to you for free and open access by the The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis) at CORE Scholar. It has been accepted for inclusion in Kno.e.sis Publications by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.
LOQUS: Linked Open Data SPARQL Querying System

Prateek Jain∗, Kunal Verma†, Peter Z. Yeh†, Pascal Hitzler∗ and Amit P. Sheth∗

∗Kno.e.sis Center, Wright State University, Dayton, OH
†Accenture Technology Labs, San Jose, CA

Abstract

The LOD cloud is gathering a lot of momentum, with the number of contributors growing manifold. Many prominent data providers have submitted and linked their data to other dataset with the help of manual mappings. The potential of the LOD cloud is enormous ranging from challenging AI issues such as open domain question answering to automated knowledge discovery. We believe that there is not enough technology support available to effectively query the LOD cloud. To this effect, we present a system called Linked Open Data SPARQL Querying System (LOQUS), which automatically maps users queries written in terms of a conceptual upper ontology to different datasets, creates a query plan, sends sub-queries to the different datasets, merges the results and presents them to the user. We present a qualitative evaluation of the system based on real-world queries posed by other researchers and practitioners.

Introduction

The Linked Open Data (LOD) methodology has recently emerged as a powerful way of linking together disparate data sources (Bizer, Heath, and Berners–Lee 2009). Using this methodology, researchers have interlinked data from diverse areas such as life sciences, nature, geography, and entertainment. Moreover, many prominent datasources (e.g. Wikipedia†, PubMed‡, data.gov§, etc.) – have also adopted this methodology to interlink their data.

The result is the LOD cloud ⁴ – a large and growing collection of interlinked public datasets represented using RDF and OWL. Concepts (and instances) in a dataset are connected to (and hence can be reached from) related concepts (and instances) from other datasets through semantic relationships such as owl:sameAs. Hence, the LOD cloud is becoming the largest currently available structured knowledge base. It has a potential for applicability in many AI-related task such as open domain question answering, knowledge discovery, and the Semantic Web.

An important prerequisite before the LOD cloud can enable these goals is allowing its users (and applications) to effectively pose queries to and retrieve answers from it. This prerequisite, however, is still an open problem for the LOD cloud. For example, in order to answer the following query from Jamendo⁵ using the LOD cloud:

Select artists within Jamendo who made at least one album tagged as ‘punk’ by a Jamendo user, sorted by the number of inhabitants of the places they are based near.

This query requires user to select the relevant datasets, identify the concepts in these datasets that the query maps to, and merge the results from each dataset into a complete answer. These steps are very costly in terms of time and required expertise which is not feasible given the size (and continued growth) of the LOD cloud. Apart from the sheer size, issues such as schema heterogeneity and entity disambiguation identified in (Jain et al. 2010) present profound challenges with respect to querying of the LOD cloud.

In this paper, we present a Linked Open Data SPARQL Querying System (LOQUS) – which allows users to effectively pose queries to the LOD cloud without having to know the exact structure and links between its many datasets. LOQUS automatically maps the user’s query to the relevant datasets (and concepts) using an upper level ontology; then executes the resulting query; and finally merges the results into a single, complete answer.

We perform a qualitative evaluation of LOQUS on several real-world queries and demonstrate that LOQUS allows users to effectively execute queries over the LOD cloud without a deep understanding of its datasets. We also compare LOQUS with existing query systems for the LOD cloud to highlight the pros and cons of each approach.

The rest of the paper is organized as follows. We begin by providing the motivation behind our work. We then introduce our approach followed by an end-to-end example and evaluation. We conclude with related work, conclusion, and future work.

Motivation

SPARQL⁶ has emerged as the de-facto query language for the Semantic Web community. It provides a mechanism to express constraints and facts, and the entities matching those constraints are returned to the user. However, the syntax of

Copyright © 2010, The authors. All rights reserved.

1http://en.wikipedia.org/wiki/Main_Page
2http://www.ncbi.nlm.nih.gov/pubmed/
3http://data.gov
4http://linkeddata.org/
5http://dbtune.org/jamendo/
6http://www.w3.org/TR/rdf-sparql-query/
SPARQL requires users to specify the precise details of the structure of the graph being queried in the triple pattern. To ease querying from an infrastructural perspective, data contributors have provided public SPARQL endpoints to query the LOD cloud datasets. But with respect to a systematic querying of the LOD cloud, we believe that the following challenges identified previously in (Jain et al. 2010) make the process difficult and should be addressed.

- **Intimate knowledge of datasets:** To formulate a query which spans multiple datasets (such as the one mentioned in the introduction) the user has to be familiar with multiple datasets. The user also has to express the precise relationships between concepts in the RDF triple pattern, which even in trivial scenarios implies browsing at least two to three datasets.

- **Schema heterogeneity:** The LOD cloud datasets cater to different domains, and thus require different modeling schemes. For example, a user interested in music related information has to skim through at least three different music related datasets such as Jamendo, MusicBrainz, MySpace. Even though the datasets belong to same domain, each have been modelled differently depending on the creator. This is perfectly fine from a knowledge engineering perspective, but it makes the querying of the cloud difficult as it requires users to understand the various heterogeneous schemas. This issue stems from the Lack of Conceptual Description of the LOD datasets.

- **Entity disambiguation:** Often the LOD cloud datasets have overlapping domains and tend to provide information about the same entity. To exemplify, both DBpedia and Geonames have information about the city of Barcelona. Although Geonames references DBpedia using the owl:sameAs property, which can confuse the user as to which is the best source to answer the query. This problem gets even more compounded when contradictory facts are reported for the same entity by different datasets. For example, DBpedia quotes the population of Barcelona as 1,615,908, whereas according to Geonames it is 1,581,595. One can argue this might be because of a difference in the notion of the city of Barcelona. But that leads to another interesting question: Is the owl:sameAs property misused in the LOD cloud?

- **Ranking of results:** In scenarios where the results of the query can be computed and returned by multiple datasets, the result which should be ranked higher for a specific query becomes an interesting and important question. As presented above, the query related to population of Barcelona can be answered by multiple datasets, but which one of them is more relevant in a specific scenario? This issue has been addressed from the perspective of popularity of datasets by considering the cardinalities and types of the relationships in (Toupikov et al. 2009), but not from the perspective of requirements with regard to a specific query.

### Our Approach

From a bird’s eyes perspective, LOQUS accepts SPARQL queries serialized by the user using concepts from an upper level ontology. LOQUS identifies the datasets and the corresponding queries to be executed on these datasets using primarily the mappings of upper level ontology to these LOD cloud datasets. This section introduces the architecture of our querying system, approach used for query execution, and the utilization of mappings for sub-query construction and the technique used for processing the results. Figure 1 illustrates the overall architecture of LOQUS.

### System Architecture

LOQUS consists of the following modules (1) Upper level ontology mapped to the domain specific LOD datasets. (2) Module to identify the upper level concepts contained in the query and perform the transformations to the LOD cloud datasets. (3) Module to split the query mapped to LOD datasets concepts into subqueries corresponding to different datasets. (4) Module to execute the queries remotely and process the results and deliver the final result to the user.

#### Upper Level Ontology

The upper level ontology has been created manually by reusing concepts from SUMO (Niles and Pease 2001) and by identifying their equivalent or subsuming concepts in the LOD cloud datasets. To demonstrate, the SUMO concept of Nation can map to different concepts belonging to the datasets of the LOD cloud such as http://dbpedia.org/ontology/Country (DBpedia), http://www.geonames.org/ontology#A.PCLI (Geonames) and http://data.linkedmdb.org/resource/movie/country (linkedmdb). These mappings are the at the schema level, and thus complement the existing mappings at the instance level provided by LOD cloud. Thus, reusing SUMO provides a single point of reference for querying the LOD cloud and consequently helps in query formulation. Further, because the mappings are at the schema level, the ontology can be utilized for reasoning and knowledge discovery over LOD cloud datasets.

#### Mapping of Upper Level Concepts to LOD Datasets

Using the mappings from SUMO, the concepts specified in the query can be mapped to concepts of the LOD cloud datasets. The concepts from LOD cloud dataset are substituted in the basic graph pattern (in lieu of concepts from SUMO) of the SPARQL query to create a query containing only concepts from the LOD datasets. The presence or absence of multiple mappings for a given concept gives an indication if the corresponding subqueries (which are created in the next step) should be involved in a union or if they should be joined to each other. Hence, this step also helps in creating a query plan for the execution and processing of results of the sub-queries.

#### Splitting of the Query Graph to Create Sub-Queries

The SPARQL query containing the concepts from the LOD cloud datasets is partitioned into sub-queries corresponding
Execution of Queries and Processing of Results The foundation of the LOD cloud is on the reuse of URIs across datasets typically to assert similarity between concepts or to link them. In order to search for concepts similar to the variables of the queries created in the previous step, their graph is appended with triples querying for "owl:sameAs", "skos:closeMatch" and similar relations using the OPTIONAL pattern of SPARQL. This step helps in identifying similar concepts and also join results from different datasets.

The results retrieved from the execution of the queries are processed according to the query plan. For example, assume that the query plan suggests that results for execution of query “Search for nations and their corresponding populations” (executed on Geonames and DBpedia), should be in a “union” with each other. To perform this operation similar concepts are identified and grouped together. The similarity is identified by using similarity properties such as "owl:sameAs" or "skos:closeMatch". Thus, the Geonames resource for Haiti http://sws.geonames.org/3723988/ can be linked to the CIA Factbook concept http://www4.wiwiss.fu-berlin.de/factbook/resource/Haiti by using the equivalence established by the DBpedia concept for Haiti http://dbpedia.org/page/Haiti, using an "owl:sameAs" link. Hence, answers from sub-queries can be merged and joined together. This mechanism also allows for finding results in scenarios which do not have a direct link by traversing some common well known similarity properties as mentioned above and retrieving information from there.

Scenario Illustration A query submitted by the user using the upper level ontology searching for "Identify films, the nations where they were shot and the population of these countries" undergoes the following process.

1. The user looks at the upper level ontology to identify the relevant concepts and serializes them into a SPARQL query.

```sparql
SELECT ?film ?nation ?pop
WHERE
  ?nation rdf:type dbpedia:Country;
  geo:featureCode geo:A.PCLI;
  dbprop:populationCensus ?pop ;
  geo:hasCountryCode geo:A.PCLI ;
  geo:population ?pop .
}
```
2. By utilizing the mappings the LOD cloud dataset specific query concepts are substituted in lieu of upper level ontology concepts.

```sparql
SELECT ?film ?nation ?pop
WHERE { ?film linkedmdb:country ?nation;
    rdf:type linkedmdb:film.
    ?nation rdf:type dbpedia:Country;
    geo:featureCode geo:A.PCLI;
    dbprop:populationCensus ?pop ;
    geo:population ?pop.}
```

3. By identifying the different datasets to which the concepts mentioned in the query graph pattern belongs, various sub-queries are created (each of which belong to a separate dataset). The query plan is also generated at this step by identifying if upper level ontology concept has multiple mappings or single mapping to LOD cloud dataset. For example, results of queries executed on datasets which provide demographic information such as DBpedia and geonames will be in "UNION", whereas LinkedMDB query results would be joined with these results.

```sparql
UNION
SELECT ?nation ?pop ?nation1 ?propertyvar
WHERE { ?nation rdf:type db:Country;
    dbprop:populationCensus ?pop.
    OPTIONAL{?nation owl:sameAs ?nation1.}
    OPTIONAL{?nation skos:closeMatch ?nation1.}
    OPTIONAL{?nation ?propertyvar ?nation1.} }
```

4. Using an available mapping of datasets and their corresponding SPARQL endpoints, the sub-queries are executed and the Table 1 to Table 3 illustrates some of the results fetched by the three sub-queries given above.

5. Finally the results of these sub-queries are processed according to the preidentified query plan. The results to be involved in UNION are merged using equivalence properties such as "owl:sameAs", whereas the query results to be in JOIN are combined by looking for similar concepts. The generated results as illustrated in Table 4 are returned to the user.

### Evaluation

As a proof of concept we have implemented LOQUS using the Jena Semantic Web Framework. The system takes a SPARQL query serialized by the user using concepts from the upper level ontology, and performs the appropriate mapping. LOQUS then executes the query and merges the results and presents the results to the user.

We perform a qualitative evaluation of our system with DARQ (Quilitz and Leser 2008) and SQUIN (Hartig, Bizer, and Freytag 2009). Our objective is to determine whether our system allows users to execute and retrieve answers to SPARQL queries over the LOD cloud without knowing the individual datasets and by just using the concepts from the upper level ontology. The lack of specification of LOD datasets in the queries requires good quality mappings to correctly identify the datasets which can be useful in answering the queries. Further, the system has to provide an efficient processing of the results for combining the results of sub-queries.

A standard measure for assessing the quality of querying systems are precision and recall. In our case, however, there does not exist any benchmarks or even available baselines for measuring these statistics partly because this is an emerging area. The sheer size of the LOD cloud makes it...
Queries and Results: To evaluate our objective we took queries which require information from multiple LOD datasets and serialized them into SPARQL queries using concepts from the upper level ontology. Table 5 presents some of the queries used for evaluating LOQUS along with statistics related to the execution of these queries. The queries though small in number require information from different sections of the LOD cloud and some of them have been adopted from publicly available sources. The queries have been executed successfully by LOQUS in a manner similar to Query 1 (which is explained in Scenario Illustration). All these queries are diverse and have different characteristics. Query 1 does not involve any concepts from LOD cloud datasets and the mentioned terms are variables or concepts from upper level ontology. Query 2 is taken from Jamendo website. In the corresponding SPARQL query, apart from URI for "Punk" taken from Jamendo, the remaining terms are again either variables or concepts from upper level ontology. Query 3 involves processing results of queries on LOD datasets (USCensus and SemWebCorpus), which do not share a direct link in the LOD cloud. Thus, LOQUS can unify answers even when sub-query answers are not directly connected to each other. Query 4 (adopted from DARQ) is identical in spirit to Query 2 as it mentions specific LOD cloud concept (From DBpedia). However, the query utilizes information from a single source. This illustrates, that LOQUS can execute and process results for queries involving just one dataset as well.

Our results demonstrate that we are able to provide a mechanism to execute challenging queries on the LOD cloud without any compromise on execution time and by covering relevant datasets. The LOQUS approach also allows queries to retrieve and merge results which involve resources not directly connected to each other in the LOD cloud. Our evaluation shows that the LOQUS approach allows effective federation of SPARQL queries over the LOD cloud by using SUMO, a common upper level ontology. Using this approach we are able to answer queries, which cannot be answered by state of the art systems for LOD query processing. Table 5 presents the various parameters on which LOQUS was evaluated for the three queries. Due to the restrictions imposed by SPARQL endpoints, the number of results returned for the query may not match the total number of entities available in datasets. The execution time has been averaged over 5 runs of the query.

Qualitative comparison with other tools: Table 6 compares LOQUS with DARQ and SQUIN on various parameters. The queries were executed for LOQUS. For other systems it is based on understanding of the capabilities of the system. DARQ (Quilitz and Leser 2008) is a query engine which provides transparent query access to multiple, distributed SPARQL endpoints as if querying a single RDF graph which relies on "Service Descriptions" to specify the capabilities of a SPARQL endpoint. One of the limitations of DARQ is the use of predicates to decide the SPARQL endpoint to send triple patterns. Hence, it requires predicates to be bound. Thus it requires use of multiple queries to fetch results for Query 1 and Query 2. Absence of direct link between SemWebCorpus and USCensus, makes it impossible to fetch results for Query 3 using DARQ. SQUIN (Hartig, Bizer, and Freytag 2009) allows LOD query answering by asynchronous traversal of RDF links to discover data that might be relevant for a query during the query execution itself. Hence, it requires at least one ground concept in the "subject" or "predicate" position of the triples contained in the query. Due to this requirement for crawling data, it is not possible to answer Query 1. Similarly Query 3 requires crawling to be performed from two different ends and then merging the crawled results and hence cannot be answered by SQUIN.

Related Work: To the best of our knowledge this is the first work presenting a system which allows users to query the LOD cloud without knowing the concepts from the diverse datasets and their interlinks. However, there are existing work on querying the LOD cloud which expects the user to know the concepts and the datasets which can answer the queries (introduced in the paper). These systems expect user to know the datasets and cannot answer the queries used for our evaluation. Another body of work which is related is the work in upper level ontology creation. A number of well known upper level ontologies such as SUMO (Niles and Pease 2001), Cyc (Reed and Lenat 2002), and DOLCE (Gangemi et al. 2002) are available. In the past various domain specific ontologies have been integrated with these upper level ontologies (Oberle and others 2007; de Melo, Suchanek, and Pease 2008) driven by application specific needs. Other bodies of work relevant for this research is in the area of federation of database queries and schema matching and mapping.

Conclusion and Future Work: We have presented an approach for querying the LOD cloud without intimate knowledge of the individual datasets and the interconnecting relationships. Our results demonstrate that we are able to provide a mechanism to execute challenging queries on the LOD cloud without any compromise on execution time and by covering relevant datasets. The LOQUS approach allows automatic retrieval and merging of results for queries involving resources indirectly linked in the LOD cloud. Our evaluation shows LOQUS approach allows effective federation of SPARQL queries over the LOD cloud by using SUMO, a common upper level ontology. Using this approach we are able to answer queries, which cannot be answered by state of the art systems for LOD query
Table 5: Result execution of queries using LOQUUS

<table>
<thead>
<tr>
<th>no.</th>
<th>query</th>
<th># results</th>
<th>Datasets</th>
<th>execution time(seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>Identify movies, countries where they were shot and the latest population for these countries.</td>
<td>1023</td>
<td>LinkedMDB, Geonames, DBpedia</td>
<td>80</td>
</tr>
<tr>
<td>Q2</td>
<td>Identify artists, whose albums have been tagged as punk and the population of the places they are based near.</td>
<td>54</td>
<td>Jamendo, MusicBrainz, Geonames, DBpedia</td>
<td>85</td>
</tr>
<tr>
<td>Q3</td>
<td>Identify congressional districts with active researchers in the area of Semantic Web.</td>
<td>30</td>
<td>SemWebCorpus, FOAF, US Census</td>
<td>110</td>
</tr>
<tr>
<td>Q4</td>
<td>Find name, birthday and image of German musicians born in Berlin.</td>
<td>8</td>
<td>DBpedia</td>
<td>65</td>
</tr>
</tbody>
</table>

Table 6: Comparison LOD SPARQL Query Processing Systems

<table>
<thead>
<tr>
<th>Metric</th>
<th>LOQUUS</th>
<th>DARQ</th>
<th>SQUIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires user to know LOD</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Datasets</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approach</td>
<td>Uses upper level ontology (SUMO) for query serialization and execution.</td>
<td>Requires formal description of datasets in the form of Service Description.</td>
<td>Requires an initial URI to execute queries.</td>
</tr>
<tr>
<td>Query Creation</td>
<td>Creates query corresponding to every mapping for a concept.</td>
<td>Creates queries only corresponding to the concepts mentioned in the query.</td>
<td>Creates queries only corresponding to the concepts mentioned in the query.</td>
</tr>
<tr>
<td>Failsafe</td>
<td>Executes all subqueries for multiple mappings. Hence retrieves at least partial answers if a specific endpoint doesn’t work.</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Result Processing</td>
<td>Query answers, retrieved from different datasets are merged and presented to user.</td>
<td>Retrieves answers from only one dataset.</td>
<td>Retrieves answers from only one dataset.</td>
</tr>
<tr>
<td>Queries Answered</td>
<td>Q1,Q2,Q3,Q4</td>
<td>Q4</td>
<td>Q2,Q4</td>
</tr>
</tbody>
</table>

Our future work includes extending the upper level ontology for other datasets, analysis of query logs for better support of query answering and optimization of query plans for faster query execution. We also plan to release the querying system and upper level ontology as an open source project. We could not provide an online querying system as demonstrator for this submission due to the required anonymity. This will be added in the final version of the paper.

References


