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[1], PubMed[2], data.gov[3], 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 Jamendo[5] 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 heterogenity 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

SPARQL[6] 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

[1]http://en.wikipedia.org/wiki/Main_Page

[2]http://www.ncbi.nlm.nih.gov/pubmed/

[3]http://data.gov

[4]http://linkeddata.org/

[5]http://dbtune.org/jamendo/

[6]http://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

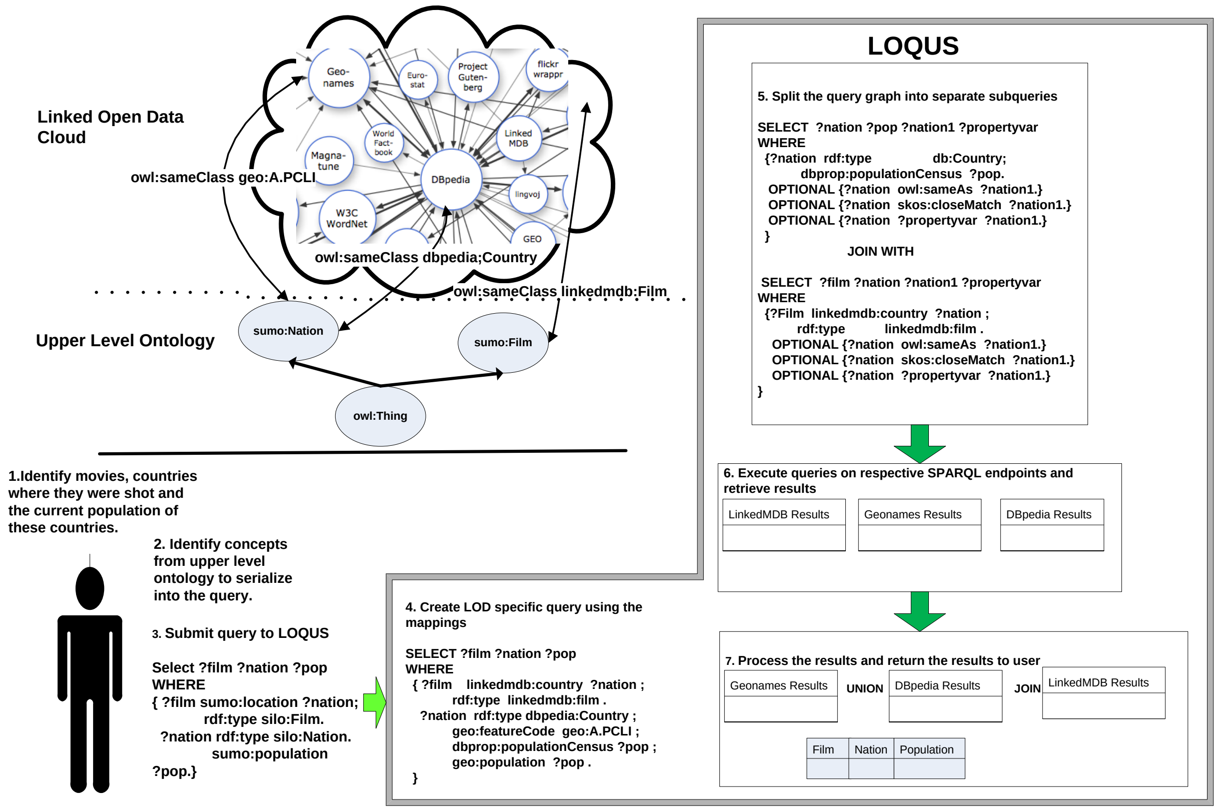
From a bird's eyes perspective, LOQUS accepts SPARQL queries serialized by the user using concepts from an upper level ontology. LOQUS identifies the datasets and the corresponding queries to be excuted on these datasets using primarily the mappings of upper level ontology to these LOD cloud datasets. 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 translations 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 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



1.Identify movies, countries where they were shot and the current population of these countries.

2. Identify concepts from upper level ontology to serialize into the query.

3. Submit query to LOQUS

```
Select ?film ?nation ?pop
WHERE
{ ?film sumo:location ?nation;
            rdf:type silo:Film.
  ?nation rdf:type silo:Nation.
              sumo:population
?pop.}
```

4. Create LOD specific query using the mappings

```
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 .
 }
```

5. Split the query graph into separate subqueries

```
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.}
 }
                  JOIN WITH

 SELECT  ?film ?nation ?nation1 ?propertyvar
WHERE
 {?Film  linkedmdb:country  ?nation ;
        rdf:type        linkedmdb:film .
   OPTIONAL {?nation  owl:sameAs  ?nation1.}
   OPTIONAL {?nation  skos:closeMatch  ?nation1.}
   OPTIONAL {?nation  ?propertyvar  ?nation1.}
}
```

6. Execute queries on respective SPARQL endpoints and retrieve results

7. Process the results and return the results to user

Figure 1: LOQUS Architecture

to the datasets whose concepts are being used in the query. The division of the original query graph is done by analyzing the namespaces of the concepts and taking cognizance of the fact, that some vocabularies such as FOAF and SIOC are reused by other datasets.

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.

```
Select ?film ?nation ?pop
WHERE
{ ?film sumo:location ?nation;
        rdf:type sumo:film.
```

```
  ?nation rdf:type sumo:Nation;
      sumo:population ?pop.}
```

2. By utilizing the mappings the LOD cloud dataset specific query concepts are substituted in lieu of upper level ontology concepts.

```
Select ?film ?nation ?pop
WHERE {?film linkedmdb:country ?nation;
         rdf:type linedmdb: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.

```
{

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.}
       }
```

UNION

```
SELECT ?nation ?pop ?nation1 ?propertyvar
WHERE {?nation geo:featureCode geo:A.PCLI;
               geo:population  ?pop.
OPTIONAL{?nation owl:sameAs ?nation1.}
OPTIONAL{?nation skos:closeMatch ?nation1.}
OPTIONAL{?nation ?propertyvar ?nation1.}
      }

}
```

JOIN

```
SELECT ?Film ?nation ?nation1 ?propertyvar
WHERE {?Film linkedmdb:country ?nation;
             rdf:type linkedmdb:film.
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.

Nation	Nation1	Population	Property Var
geo:102358	db:Saudi_Arabia	28161000	owl:sameAs
geo:1036973	db:Mozambique	21284000	owl:sameAs
geo:1269750	db:India	1147995000	owl:sameAs

Table 1: Result execution of queries over geonames

Nation	Nation1	Population	Property Var
db:Saudi_Arabia	geo:102358	28,686,633	owl:sameAs
db:Saudi_Arabia	cyc:en/SaudiArabia	28,686,633	owl:sameAs
db:Mozambique	cyc:en/Mozambique	21,397,000	owl:sameAs
db:Mozambique	umbel:Mozambique	21,397,000	owl:sameAs

Table 2: Result execution of queries over dbpedia

Film	Nation	Nation1	Property Var
lmdb:30356	lmdb:IN	geonames:1269750	skos:closeMatch
lmdb:27302	lmdb:SA	geonames:102358	skos:closeMatch
lmdb:35434	lmdb:MZ	geonames:1036973	skos:closeMatch

Table 3: Result execution of queries over linkedmdb

Film	Nation	Population	Nation	Population
lmdb:30356	geo:1269750	1147995000	db:India	1028610328
lmdb:27302	geo:102358	28161000	db:Saudi_Arabia	28,686,633

Table 4: Result of user submitted query

Evaluation

As a proof of concept we have implemented LOQUS using the Jena[7] 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

[7]http://jena.sourceforge.net/

difficult to identify if all correct answers have been retrieved and reported. Currently there is no easy way to create a baseline for a large set of LOD cloud queries because there are no available systems which can perform the task in a complete manner, as required for creating a baseline reference. At the same time, SPARQL endpoints also restrict the number of results returned for a specific query. Hence, getting complete sets of answers is a challenge.

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 involves 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 federatation 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 other 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

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

Table 5: Result execution of queries using LOQUS

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

Table 6: Comaparison LOD SPARQL Query Processing Systems

processing.

Our future work includes extending the upper level ontology for including 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

Bizer, C.; Heath, T.; and Berners-Lee, T. 2009. Linked data – the story so far. *IJSWIS* 5(3):1–22.

de Melo, G.; Suchanek, F.; and Pease, A. 2008. Integrating YAGO into the Suggested Upper Merged Ontology. In *ICTAI 2008*.

Gangemi, A.; Guarino, N.; Masolo, C.; Oltramari, A.; and Schneider, L. 2002. Sweetening Ontologies with DOLCE. In *EKAW 2002*, volume 2473 of *LNCS*, 223–233.

Hartig, O.; Bizer, C.; and Freytag, J.-C. 2009. Executing SPARQL Queries over the Web of Linked Data. In *ISWC 2009*, volume 5823 of *LNCS*, 293–309.

Jain, P.; Hitzler, P.; Yeh, P. Z.; Verma, K.; and Sheth, A. P. 2010. Linked Data is Merely More Data. *AAAI Spring Symposium "Linked Data Meets Artificial Intelligence"*.

Niles, I., and Pease, A. 2001. Towards a Standard Upper Ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems – Volume 2001*, 2–9.

Oberle, D., et al. 2007. DOLCE ergo SUMO: On Foundational and Domain Models in the SmartWeb Integrated Ontology (SWIntO). *JWS* 5(3):156–174.

Quilitz, B., and Leser, U. 2008. Querying Distributed RDF Data Sources with SPARQL. In *ESWC 2008*, volume 5021 of *LNCS*, 524–538.

Reed, S., and Lenat, D. 2002. Mapping Ontologies into Cyc. Technical report, Cycorp, Inc,. Available from http://www.cyc.com/doc/white_papers/.

Toupikov, N.; Umbrich, J.; Delbru, R.; Hausenblas, M.; and Tummarello, G. 2009. DING! Dataset ranking using formal descriptions. In *WWW2009 Workshop on Linked Data on the Web (LDOW2009)*.