# Alignment-based Querying of Linked Open Data

Amit Krishna Joshi[1], Prateek Jain[1], Pascal Hitzler[1], Peter Z. Yeh[2], Kunal Verma[2], Amit P. Sheth[1], and Mariana Damova[3]

[1] Kno.e.sis Center, Wright State University, Dayton, OH, U.S.A.
[2] Accenture Technology Labs San Jose, CA, U.S.A.
[3] OntoText, Bulgaria

**Abstract.** The Linked Open Data (LOD) cloud is rapidly becoming the largest interconnected source of structured data on diverse domains. The potential of the LOD cloud is enormous, ranging from solving challenging AI issues such as open domain question answering to automated knowledge discovery. However, due to an inherent distributed nature of LOD and a growing number of ontologies and vocabularies used in LOD datasets, querying over multiple datasets and retrieving LOD data remains a challenging task. In this paper, we propose a novel approach to querying linked data by using alignments for processing queries whose constituent data come from heterogeneous sources. We also report on our Alignment based Linked Open Data Querying System (ALOQUS) and present the architecture and associated methods. Using the state of the art alignment system BLOOMS, ALOQUS automatically maps concepts in users' SPARQL queries, written in terms of a conceptual upper ontology or domain specific ontology, to different LOD concepts and datasets. It then creates a query plan, sends sub-queries to the different endpoints, crawls for co-referent URIs, merges the results and presents them to the user. We also compare the existing querying systems and demonstrate the added capabilities that the alignment based approach can provide for querying the Linked data.

## 1 Introduction

Linked Open Data (LOD) has recently emerged as a powerful way of linking together disparate data sources [3]. A community of contributors have collected and interlinked over 30 billion facts[4] from diverse areas such as life sciences, nature, geography, media and entertainment. Prominent data publishers such as The New York Times,[5] the US government,[6] the UK government,[7] BBC Music,[8] and PubMed[9] have also adopted this methodology to interlink their data. The

[4] http://www4.wiwiss.fu-berlin.de/lodcloud/state/
[5] http://data.nytimes.com/home/
[6] http://data.gov
[7] http://data.gov.uk/data
[8] http://www.bbc.co.uk/music
[9] http://www.ncbi.nlm.nih.gov/pubmed/

result is the LOD cloud[10]—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 with data about music, movies, reviews, scientific publications, government information, geographical locations, medicine and many more. It has a potential for applicability in many AI-related tasks such as open domain question answering, knowledge discovery, and the Semantic Web. However, to take advantage of the enormously extensive structured data in the LOD cloud, one must be able to effectively pose queries to and retrieve answers from it.

However, querying the LOD cloud is still a challenge as it requires users to understand various concepts and datasets prior to creating a query. For example, consider the query "*Identify films, the nations where they were shot and the populations of these countries.*" Answering this query requires a 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 practical given the size (and continued growth) of the LOD cloud. Furthermore, issues such as schema heterogenity and entity disambiguation identified in [16] present profound challenges with respect to querying of the LOD cloud. Each of these data sources can be queried separately, most often through an end point using the SPARQL query language [26]. Looking for answers making use of information spanning over different data sets is a more challenging task as the mechanisms used internally to query datasets (database-like joins, query planning) cannot be easily generalized to this setting.

In this paper, we present a novel approach towards querying of linked data across multiple datasets and report on our Alignment based Linked Open Data SPARQL Querying System (ALOQUS) which allows users to effectively pose queries to the LOD cloud without having to know the representation structures or the links between its many datasets. ALOQUS automatically maps the user's query to the relevant datasets (and concepts) using state of the art alignment methods; then executes the resulting query by querying each of the datasets separately; and finally merges the results into a single, complete answer. We perform a qualitative evaluation of ALOQUS on several real-world queries and demonstrate that ALOQUS allows users to effectively execute queries over the LOD cloud without a deep understanding of its datasets. We also compare ALOQUS with existing query systems for the LOD cloud to highlight the pros and cons of each approach.

The paper is organized as follows. We begin by providing the motivation behind our work in Section 2. We then introduce our approach in Section 3, followed by an end-to-end example and an evaluation, in Section 4. We briefly

[10] http://linkeddata.org/

discuss scalability issues in Section 5, before we conclude with related work, conclusions, and future work.

*Acknowledgments.* This work was supported by the National Science Foundation under award 1143717 "III: EAGER – Expressive Scalable Querying over Linked Open Data." Mariana Damova acknowledges support under the project RENDER FP7-ICT-2009-5, contract no. 257790.

## 2 Motivation

SPARQL [26] 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 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, some of which are identified previously in [16], make the process difficult and need to 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 many music related datasets such as Jamendo,[11] MusicBrainz,[12] and BBC Music. Even though the datasets belong to the same domain, they have been modeled differently depending on the creator. This is perfectly fine from a knowledge engineering perspective, but it makes querying of the LOD cloud difficult as it requires users to understand the heterogeneous schemas.

*Entity Co-reference*: The purpose of entity co-reference is to determine if different resources refer to the same real world entity [30]. Often the LOD datasets have overlapping domains and tend to provide information about the same entity [12]. The similarity is identified by using *similarity properties* such as "owl:sameAs" or "skos:exactMatch." For instance, LinkedMdb provides information about the "Romeo &Juliet" movie and provides direct reference to DBPedia using the *owl:sameAs* property. However, there are cases where the two instances might not be directly connected but a path exists for such a co-reference as shown in Figure 1. Here, the Geonames resource for China is linked to the CIA Factbook concept and the DBPedia concept for China, using an "owl:sameAs" link from the NYTimes dataset. Finding results in scenarios which

[11] http://dbtune.org/jamendo
[12] http://dbtune.org/musicbrainz/

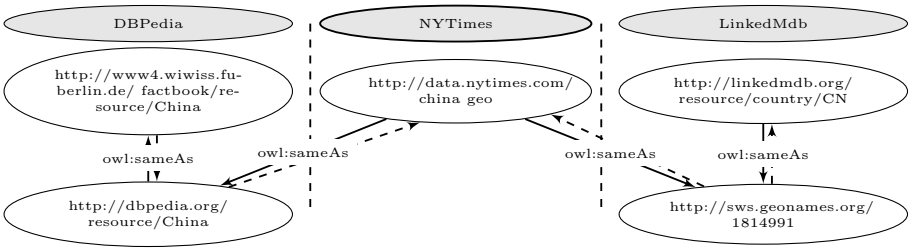


**Fig. 1.** LinkedMdb connects to DBPedia via NYTimes

do not have a direct link is thus possible by traversing some common well-known similarity properties and retrieving information from multiple datasets.

## 3 Our Approach

ALOQUS accepts SPARQL queries serialized by the user using concepts from an upper level ontology (the *primary* ontology for phrasing queries) such as PROTON [31]. ALOQUS identifies the datasets for each concept and federates sub-queries to be executed on these datasets primarily using mappings between the upper level ontology and the LOD cloud datasets. This section introduces the architecture of our querying system, the approach used for query execution, how we use mappings for constructing sub-queries, and the technique used for processing and integrating the results.

### 3.1 System Architecture

ALOQUS consists of several modules for the following purposes. (1) Automatic mapping between the upper level ontology and specific ontologies in LOD datasets. (2) Identification and mapping of concepts in user defined queries to those in LOD Datasets. (3) Constructing sub-queries. (4) Sub-query execution. (5) Determining entity co-references. (6) Transformation of RDF graphs and local storage of the sub-query results. (7) Local querying and delivery of the final result to the user.

*Automatic mapping between upper level Ontology and Ontologies used in LOD Datasets* To create an automatic mapping between the upper level ontology and ontologies used in LOD Datasets, we use the BLOOMS ontology alignment system [15, 17]. The choice of BLOOMS over other ontology alignment systems such as [7, 11, 19] is mainly due to its higher precision and recall on LOD datasets, as shown in [15]. The mappings provided by BLOOMS are at the schema level and thus complement the existing mappings at the instance level provided by the LOD cloud. Thus, reusing upper level ontologies like PROTON and SUMO [24] 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. In addition to the automatically generated mappings, we use the existing mappings used in [4] and those already available on the web.[13,14] Our system is designed with pluggable architecture and hence can use output from any Alignment System that provides mappings in the Alignment API's Alignment format [6].

*Identification and mapping of concepts in user defined queries to those in LOD Datasets* Using the mappings between an upper level ontology and other ontologies in the LOD datasets, the concepts specified in the query can be mapped to concepts of the LOD cloud datasets. Since the output of the alignment system, BLOOMS, is in the Alignment API format, the number of mappings can be restricted by providing a corresponding confidence threshold (the confidence value is a number between 0 and 1 that reflects the confidence of the system in the determined mapping [6]). For instance, the mapping from "proton:school" to DBPedia for a threshold of 1 results in a mapping to "dbpedia:school" only, but for threshold of 0.9, we get additional mappings, for example to "dbpedia:EducationalInstitution." BLOOMS suggest using a confidence value of 0.6 or higher but we found out that the number of mappings produced is often too many for our purpose so we restricted them to top $k$ (variable) mappings that meet a threshold of 0.9.

*Constructing Sub-queries* The concepts from the upper level ontology in a query are then substituted by mapped concepts to create multiple sub-queries. Each sub-query is created based on the concepts present in the corresponding datasets and taking cognizance of the fact, that some vocabularies such as FOAF, RDF and SIOC are reused by other datasets. Each of the sub-queries uses SPARQL CONSTRUCT (with upper level concepts in the graph template) instead of the SELECT query form to return an RDF graph containing triples with upper level concepts. The CONSTRUCT query form provides a mechanism to create new sets of triples, thereby making implicit LOD information explicit.

*Execution of sub-Queries* For each sub-query, a graph is constructed by querying corresponding endpoints. For instance, a sub-query containing a statement with Music Ontology[15] concepts is queried to both BBC Music[16] and Jamendo endpoints. Source selection can be done either by specifying a local metadata file [27] or by sending a SPARQL ASK query for each triple pattern to every possible endpoint [29]. For ALOQUS, we built a metadata file containing a list of endpoints, each mapped to ontologies used for the mapping. Information about vocabulary and endpoints are obtained from the CKAN directory.[17] In addition,

[13] http://www4.wiwiss.fu-berlin.de/bizer/r2r/examples/ DBpediaToX.ttl
[14] http://code.google.com/p/umbel/source/ browse/trunk/v100/External+Ontologies/
[15] http://musicontology.com/
[16] http://api.talis.com/stores/bbc-backstage/services/sparql
[17] http://thedatahub.org/

we consumed SPARQL services from Mondeca Labs' LOV endpoints[18] for vocabularies and endpoints. It should be noted that the returned graph contains triples with upper level concepts and LOD entities since upper level concepts are included in the CONSTRUCT graph template.

*Determining entity co-references* The foundation of the LOD cloud is on the reuse of URIs across datasets, typically to assert similarity between entities or to link them. In order to search for entities similar to the variables of the queries created in the previous step, we use a crawling approach that detects the additional entities through owl:sameAs and skos:exactMatch. The crawling is required because two entities might not be directly connected but via other similar entities as exemplified in Section 2 above. A query used for fetching similar entities resembles the following.

```
SELECT  ?sameAs ?property_var
WHERE
    {  {  { dbpedia:Hawaii owl:sameAs ?sameAs    }
          union        { ?sameAs owl:sameAs dbpedia:Hawaii    }
          union        { dbpedia:Hawaii skos:exactMatch ?sameAs    }
          union        { ?sameAs skos:exactMatch dbpedia:Hawaii    }  }
       optional  {     { dbpedia:Hawaii ?property_var ?sameAs      }
                       union { ?sameAs ?property_var dbpedia:Hawaii } } }
```

A simple crawling approach used in ALOQUS is described below. For each entity retrieved from a sub-query, a new query is constructed using owl:sameAs and skos:exactMatch (see above) and then queried to multiple endpoints. Following an iterative approach, it fetches similar entities and inserts them into a Set. The final result for each entity is a unique list of similar entities which are then stored in a database under a unique identifier created on the fly (eg: `http://www.knoesis.org/aloqus/uid`). The creation of such a unique identifier greatly helps for querying in subsequent steps when join needs to be performed. We call them *proxy identifiers* and a set of similar entities corresponding to each proxy identifier a *Similarity Set*. The steps can be summarized as follows.

```
Get list of entities by executing a sub-query.
For each entity, construct a new query using owl:sameAs and skos:exactMatch (as shown above).
Query to each endpoint and fetch the similar entities.
Store the entities in a Similarity Set.
For each entity in a Similarity Set which has not yet been queried, repeat steps 3.1 to 3.1.
Merge the constructed sets if required.
```

In addition to our own crawling approach, we consume REST services from the sameAs.org website[19] for getting equivalent URIs. It currently has over 100M URIs and returns back URIs which are co-referents for a given URI. It uses many predicates (ex: skos:exactMatch, cyc:similarTo) besides owl:sameAs to determine co-referent URIs from a variety of sources including DBPedia, NYTimes, UMBEL, OpenCyc and BBC Music. Using both of the mentioned approaches provides a larger (and hence more complete) set of similar entities and helps in identifying similar entities which do not have a direct link. We have presented

[18] http://labs.mondeca.com/endpoint/lov/
[19] http://www.sameas.org/

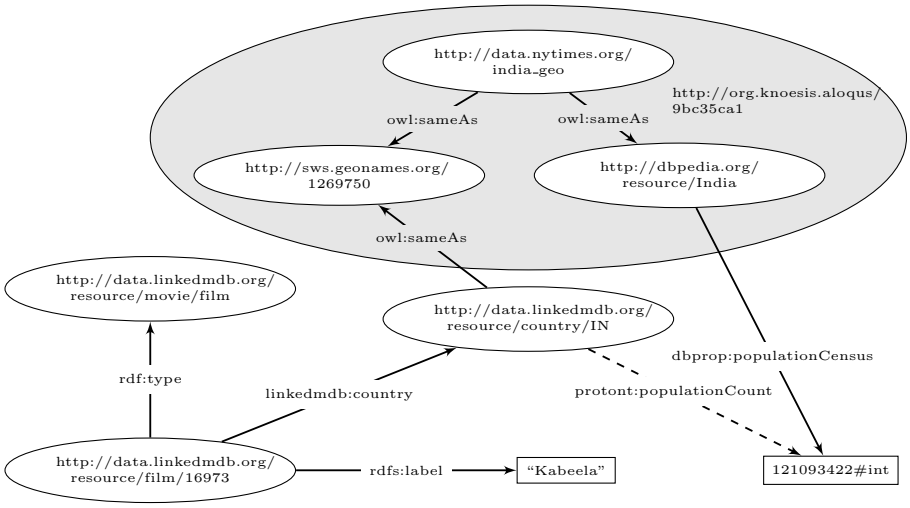


**Fig. 2.** ALOQUS Illustration

a naive way to crawl for similar entities but the system gets better as we generate more proxy identifiers and add to our database. This step is an important step for ALOQUS as it enables using common identifiers for join operations, if required in a query.

*Transformation and local storage of RDF graphs* The RDF graphs returned by the execution of sub-queries are transformed into new RDF graphs by replacing the values of variables with the proxy identifiers created during the process of entity co-reference detection. The transformed graphs are then stored to an RDF store. In addition, the mappings between each proxy identifier to corresponding similar LOD entities are also stored. The inclusion of newly created proxy identifiers in a local RDF store is important because it eventually allows us to treat our RDF store as an independent dataset and thus to perform the join operation required for the queries.

*Joining and Processing of results* With all the results from sub-queries now stored in the local RDF store, the next step is to perform an original query on the latter. It should be noted that join operations, if required in the query, would be automatically done since we have transformed all the triples to use proxy identifiers for the values of shared variables. The results can be considered final but the results include the values of variables represented in proxy identifiers. Since the mappings from proxy identifiers to values of variables returned from sub-queries are available in the datastore, all we need is to expand the result and present it to the user.

### 3.2 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    protonu:ofCountry[20]        ?nation.
        ?film    rdf:type                     protonu:Movie[21].
        ?film    rdfs:label                   ?film_name.
        ?nation  protont:populationCount[22]  ?pop.                }
```

2. By utilizing the BLOOMS mappings and getting the best alignment ($k = 1$) for each of the concepts, a set of sub-queries is generated where LOD cloud dataset specific concepts are substituted in lieu of upper level ontology concepts.

(a)
```
SELECT ?film ?nation ?pop
WHERE { ?film  lmdb:country  ?nation.
        ?film  rdf:type      lmdb:film.
        ?film  rdfs:label    ?film_name.       }
```

(b)
```
SELECT ?nation ?pop
WHERE { ?nation  dbprop:populationCensus ?pop.   }
```

3. The subqueries are then executed in the corresponding end-points. Both the above sub-queries are transformed to use the SPARQL CONSTRUCT query form so that we get the graph instead of a table of results. It should be noted that the CONSTRUCT clause uses concepts from the upper level ontologies. For instance, the sub-query 2a is converted to

```
CONSTRUCT { ?film  protonu:ofCountry ?nation.
            ?film  rdf:type          protonu:Movie.
            ?film  rdfs:label        ?film_name.       }
WHERE { ?film  lmdb:country  ?nation.
        ?film  rdf:type      lmdb:film.
        ?film  rdfs:label    ?film_name.   }
```

4. Some triples from the returned graphs (in Turtle format) are shown below. This includes triples with LOD entities and upper level concepts.

```
lmdb-film:11446   protonu:ofCountry  lmdb-country:IN.
lmdb-film:11446   rdf:type           protonu:Movie.
lmdb-film:11446   rdfs:label         "Run".
lmdb-film:17091   protonu:ofCountry  lmdb-country:LK.
lmdb-film:17091   rdf:type           protonu:Movie.
lmdb-film:17091   rdfs:label         "Getawarayo".
lmdb-film:16973   protonu:ofCountry  lmdb-country:IN.
lmdb-film:16973   rdf:type           protonu:Movie.
lmdb-film:16973   rdfs:label         "Kabeela".
dbpedia:Sri_Lanka      protont:PopulationCount 21324791.
dbpedia:Czech_Republic protont:PopulationCount 10230060.
dbpedia:India          protont:PopulationCount 1210193422.
```

[20] http://proton.semanticweb.org/2005/04/protonu#ofCountry
[21] http://proton.semanticweb.org/2005/04/protonu#Movie
[22] http://proton.semanticweb.org/2005/04/protont#PopulationCount

5. By looking at the above partial results, we can find that two results can be merged (treating dbpedia:India same as lmdb-country:IN). However, the lack of common identifiers keeps the triples from two results separate. The next step is to crawl and find out the similar entities. By using the entity co-reference detection process explained earlier, some of the similar entities from the similarity set of lmdb-country:IN and lmdb-country:LK include

   ```
   http://data.linkedmdb.org/resource/country/IN
   http://sws.geonames.org/1269750/
   http://rdf.freebase.com/ns/m.03rk0
   http://dbpedia.org/resource/India
   http://data.nytimes.com/india_geo
   http://dbtune.org/musicbrainz/resource/country/IN
   http://umbel.org/umbel/ne/wikipedia/India
   http://www.ontologyportal.org/SUMO.owl#India
   http://www4.wiwiss.fu-berlin.de/factbook/resource/India
   ```

   and

   ```
   http://data.linkedmdb.org/resource/country/LK
   http://rdf.freebase.com/ns/m.06m_5
   http://dbpedia.org/resource/Sri_Lanka
   http://data.nytimes.com/sri_lanka_geo
   http://lexvo.org/id/iso3166/LK
   http://linkedgeodata.org/triplify/node424311565
   http://mpii.de/yago/resource/Sri_Lanka
   http://psi.oasis-open.org/iso/3166/#144
   http://sw.opencyc.org/2008/06/10/concept/en/SriLanka
   ```

   respectively.
6. The proxy identifiers and similarity sets are created at the same step resulting, e.g., in aloqus:2908ba82 and aloqus:9bc35ca1 identifiers for all the items in the similarity set of lmdb-country:LK and lmdb-country:IN, respectively.
7. The RDF graphs returned by the execution of sub-queries are transformed to include only the proxy identifiers for all the values of the variables that are shared among multiple statements in the original query. The variable $film$ need not have the proxy identifiers but the *nation* should, since it is used in more than one statement. In essence, we are looking for common identifiers that would aid in the join operation.

   ```
   lmdb-film:11446 rdfs:label         "Run".
   lmdb-film:11446 protonu:ofCountry  aloqus:9bc35ca1.
   lmdb-film:11446 rdf:type           protonu:Movie.
   lmdb-film:17091 rdfs:label         "Getawarayo".
   lmdb-film:17091 protonu:ofCountry  aloqus:2908ba82.
   lmdb-film:17091 rdf:type           protonu:Movie.
   aloqus:2908ba82 protont:populationCount 21324791.
   aloqus:9bc35ca1 protont:populationCount 1210193422.
   ```

8. The transformed graphs are stored in a local RDF store and an original query is executed on it to fetch the results. The intermediate and final results are shown in Tables 1 and 2.

## 4 Evaluation

As a proof-of-concept evaluation for our alignment based approach towards querying of Linked Open Data, an ALOQUS prototype has been implemented

**Table 1.** Result containing proxy identifiers

| film | name | nation | population |
|---|---|---|---|
| lmdb-film:17091 | "Getawarayo" | aloqus:2908ba82 | 21324791 |
| lmdb-film:16973 | "Kabeela" | aloqus:9bc35ca1 | 1210193422 |
| lmdb-film:11446 | "Run" | aloqus:9bc35ca1 | 1210193422 |

**Table 2.** Result containing LOD identifiers

| film | name | nation | population |
|---|---|---|---|
| lmdb-film:17091 | "Getawarayo" | lmdb-country:LK | 21324791 |
| lmdb-film:16973 | "Kabeela" | lmdb-country:IN | 1210193422 |
| lmdb-film:11446 | "Run" | lmdb-country:IN | 1210193422 |
| lmdb-film:11446 | "Run" | nytimes:india_geo | 1210193422 |

using the Jena[23] Semantic Web Framework. The system takes a SPARQL query serialized by the user using concepts from the upper level ontology, and generates the appropriate mappings. For our purposes, we generated mappings between PROTON and various LOD ontologies including DBPedia, Music Ontology, LinkedMdb, the BBC Programme Ontology,[24] Factbook[25] and Semantic Web Corpus.[26] These mappings are generated only once and additional mappings can be generated and added at any later time. ALOQUS then generates multiple sub-queries, executes them and crawls for co-referent URIs before merging the results and presenting the results to the user. The intermediate results are stored in a local TDB Store.[27]

A standard measure for assessing the quality of querying systems are precision and recall. In our case, however, there does not exist any benchmark nor are there any baselines available for measuring these statistics partly because not much work has been done in alignment based query processing systems. Furthermore, the sheer size of the LOD cloud and its continuing growth makes it difficult to identify if all correct answers have been retrieved and reported. For these reasons, we present a test harness consisting of three different query types (discussed in Section 4.1) that can be used for evaluating ALOQUS and similar systems that will be developed by researchers in our community in the future. We will propose a future evolution of this test harness through the Ontology Alignment Evaluation Initiative(OAEI).[28]

We also performed a qualitative evaluation of our system by comparing it with DARQ [27] and SQUIN [13]. Systems like Factforge [2] are not used for comparison because they can be considered as working on a single dataset created by assembling multiple independent datasets. Our objective is to determine whether

[23] http://jena.sourceforge.net/
[24] http://www.bbc.co.uk/ontologies/programmes/2009-09-07.shtml
[25] http://www.daml.org/2003/09/factbook/factbook-ont
[26] http://data.semanticweb.org/ns/swc/swc_2009-05-09.rdf
[27] http://openjena.org/TDB/
[28] http://oaei.ontologymatching.org

our system allows users to execute and retrieve answers to SPARQL queries over the LOD cloud without intimate knowledge of individual datasets and by using 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. We show that our reliance on BLOOMS, a state of the art alignment system,provides adequate answers to our queries.

### 4.1 Query Types

In this section, we introduce several terms for classifying queries that any alignment based querying system can be evaluated on with respect to a collection of datasets. To differentiate different query types, we introduce three types of query statements viz., *Akin Statement*, *Alien Statement* and *Allied Statement*.

A statement $S$ occurring in a query $Q$ is classified as an *Akin Statement* if all the predicates (concepts or properties) mentioned in the statement belong to the reference set of LOD ontologies. On the other hand, a query statement is an *Alien Statement* if none of the concepts and properties mentioned in the statement can be found in ontologies in the reference set (for instance, a statement containing terms from the upper level ontology only). An *Allied Statement* is one which has a combination of predicates, at least one existent and one non-existent in the reference set of ontologies. This type of query statement is of particular importance since the user has partial knowledge of the expected triples. The notion of Akin Statement generally refers to the connected statements that are already present in the reference datasets. Based on these statement types, the following query types are introduced:

- Domestic Query: A query containing only Akin Statements.
- Foreign Query: A query containing only Alien Statements.
- Hybrid Query: A query containing a combination of different statement types.

Each of the query types has a different level of complexity with respect to the required number of combinations of mappings, detection of equivalent URIs and the query federation. Domestic Queries do not need mappings and hence require only query federation and joins. Both Foreign and Hybrid queries involve predicate mappings in addition to federation and joins to fetch the results. Queries containing Alien statements can lead to a huge number of mappings and require both crawling and federation to a large number of endpoints. It should be noted that execution of Foreign queries within the reference datasets will always return an empty result set since the relevant concepts and properties do not occur in any triples in these datasets.

We further declare a set $V$ of vocabularies, whose appearance in the query statement should be ignored for classifying statement types. This flexibility is provided taking into consideration the fact that certain vocabularies such as RDF and FOAF have ubiquitous presence and are often required even when a user wants to use only upper level ontologies.

| no. | Query | Datasets | Primary Ontology | Other LOD Ontologies | Query Type |
|---|---|---|---|---|---|
| Q1 | Identify movies, countries where they were shot and the latest population for these countries. | LinkedMDB, DBpedia | PROTON | N/A | Foreign |
| Q2 | List the semantic web people and their affiliation. | Semantic Web Dog Food | N/A | SWRC | Domestic |
| Q3 | Find all Jamendo artists along with their image, home page, and the population of city they are near. | Jamendo, Geonames | N/A | Music Ontology, Geonames | Domestic |
| Q4 | Software companies founded in the US | DBpedia | PROTON | DBPedia | Hybrid |
| Q5 | Find list of movies, director and actors and the population of their birth cities. | DBpedia, LinkedMdb, Factbook | PROTON | LinkedMdb | Hybrid |
| Q6 | List the countries, birth rates and sex ratios. | DBPedia, Factbook | PROTON | Factbook | Hybrid |
| Q7 | Is Mayotte a country? | DBPedia | PROTON | N/A | Foreign |
| Q8 | Get the birthdates of folks who acted in Star Trek | DBPedia, LinkedMdb | PROTON | N/A | Foreign |
| Q9 | List Music artists and birth dates. | DBPedia, BBC Music, Jamendo | DBpedia | N/A | Domestic |
| Q10 | Find list of movies made in countries with population greater that 1 Billion. | DBpedia,LinkedMdb | DBPedia | N/A | Domestic |

**Table 3.** ALOQUS Queries

*Queries and Results* For evaluation purposes, we created queries of different types which require information from multiple LOD datasets, and serialized them into SPARQL queries using concepts from the upper level ontology. Table 3 presents some of the queries used for evaluating ALOQUS. 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. Here, we specify reference ontologies to be those which require mapping to be performed before generating sub-queries.

The queries have been executed successfully by ALOQUS in a manner similar to Query 1 which was illustrated in Section 3.2. Query 1, of type Foreign, does not involve any concepts from LOD cloud datasets and the mentioned terms are properties or concepts from the upper level ontology. This involves the processing of results of queries on LOD datasets, which do not share a direct link in the LOD cloud. Thus, ALOQUS can unify answers even when sub-query answers are not directly connected to each other. Query 2, of type Domestic, has been obtained from the Semantic Web Dog Food website[29] and does not require any mappings to be performed. Query 3 is another example of type Domestic but requires querying multiple datasets (Jamendo, Geonames) to get the results. Query 4 (adopted from FactForge), of type Hybrid, contains concepts and properties from both the upper level ontology and from LOD datasets, and hence requires mappings for some property and concepts from the upper level ontology. Queries

[29] http://data.semanticweb.org/

| Features | ALOQUS | DARQ | SQUIN |
|---|---|---|---|
| Approach | Uses upper level ontology(PROTON) or any other ontology as primary ontology 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 sub-queries for multiple mappings. Hence retrieves at least partial answers if a specific endpoint doesn't work. | X | X |
| Detect Entity co-references | Crawls and also consumes sameAs.org webservices. | X | X |
| Result Processing | Query answers, retrieved from different datasets are merged and presented to user. | Retrieves answers from multiple dataset based on service description. | Retrieves answers from multiple dataset through link traversal. |
| Write queries using ontology not present in LOD | Yes | X | X |
| Support for open-ended queries like ?s ?p ?o | Yes | X | X |
| Result Storage for later Retrieval | Yes | X | X |
| DESCRIBE Query Form | Yes | N/A | Yes |

**Table 4.** Comparison of LOD SPARQL Query Processing Systems

Q5 to Q8 are a few more Hybrid and Foreign queries. As can be seen from Table 3, ALOQUS can execute and process queries involving one or multiple datasets. Queries Q9 and Q10 show the extended capabilities of ALOQUS and will be discussed in Section 5.

Our results demonstrate that we are able to provide a mechanism to execute queries on the LOD cloud without relevant datasets' concepts in the query. The ALOQUS 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 ALOQUS approach allows effective federation of SPARQL queries over the LOD cloud by using PROTON, 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.

*Qualitative comparison with other tools* Two of the current systems, DARQ and SQUIN, which can partially answer some of the queries ALOQUS can, are compared on various metrics including query creation and entity co-reference

detection as shown in Table 4. The queries were executed for ALOQUS. For other systems it is based on an understanding of the capabilities of the system. DARQ [27] 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 which to send triple patterns. Thus it requires the use of multiple queries to fetch results for queries of type Hybrid and Foreign. The absence of a direct link between different datasets often makes it impossible to fetch results for DARQ (queries similar to Q1). SQUIN 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 able to answer queries of both the Hybrid and Foreign types which include predicates not present in the existing datasets. Both DARQ and SQUIN are expected to fetch results for Domestic queries.

## 5 Scalability Considerations

Since ALOQUS is an alignment based querying system, there is no need to limit it to using only an upper level ontology as the primary ontology for phrasing the queries. This caters for cases where the user wants to query concepts that are not in the upper level ontology but exist in some LOD dataset, or if the user wants to use a different primary ontology such as DBPedia and use ALOQUS to get additional LOD data. A user also may have a proprietary ontology to be used for phrasing queries.

Since it is impossible to create one unique ontology that can map to every other LOD dataset, ALOQUS is designed to accommodate such alternative settings. The pluggable architecture of ALOQUS allows users to use any other upper ontology or LOD ontology as a primary ontology provided that a mapping can be generated (or provided) between the chosen primary ontology and other LOD ontologies.

To validate this capability, we have tested ALOQUS using DBPedia as an alternative primary ontology. We used mappings from DBPedia to LinkedMdb and Music Ontology for Queries Q9 and Q10 in Table 3. The queries were written using only DBPedia predicates but multiple mappings were generated for the concepts defined in the query. The sub-queries were generated and queried to multiple endpoints, followed by detection of equivalent URIs and merging of the results of execution of sub-queries.

While we presented an implementation that uses our state of the art alignment system, BLOOMS, it has a flexible architecture and can use any other alignment system that might perform better in specific domains. One of the key strengths of ALOQUS architecture is that it enables automation of all the steps involved in query processing. ALOQUS is still a working prototype and lots of

enhancement can be done with better optimization techniques. At present, the equivalent URI detection phase takes longer than the rest as a large number of crawling is performed for generating proxy identifiers and Similarity Sets.

The pluggable architecture, which enables the easy use of other primary ontologies and of other alignment systems and available mappings, means that ALOQUS can be modified for different purposes, and will gain in strength as further ontologies, mappings, and alignment systems become available. ALOQUS thus scales in the sense that it can easily improve as more data and tools for dealing with LOD datasets become available.

## 6 Related Work

To the best of our knowledge, this is the first work investigating an alignment based querying of linked data allowing users to write query statements using concepts and properties not present in the LOD cloud. However, there is existing work on query federation which assumes that the user intimately knows concepts and datasets beforehand [27, 13, 18, 29]. [14] discusses a database perspective for querying Linked Data on the web including query federation, while [29] investigates optimizing techniques for federated query processing on Linked Data. ALOQUS also uses federation techniques to query distributed datasets once the sub-queries are generated. Systems like OBSERVER [23] have shown that the use of brokering across domain ontologies provides a scalable solution for accessing heterogeneous, distributed data repositories.

Work on ontology alignment and mapping [8, 9, 5, 22] provides a foundation to our approach. Since ALOQUS uses an alignment system to generate subqueries and then perform federation, any future improvement in state of the art alignment systems will also improve ALOQUS.

Another body of work which is related is work on upper level ontology creation. A number of well known upper level ontologies such as SUMO [24], Cyc [28], and DOLCE [10] are available [21]. In the past various domain specific ontologies have been integrated with these upper level ontologies [25] driven by application specific needs. FactForge [2] uses mappings between the upper level ontology PROTON and other ontologies to build a compound dataset comprising some of the most popular datasets of the LOD Cloud, e.g., DBPedia, MusicBrainz, New York Times, Freebase, Geonames and Wordnet. Systems like PowerAqua [20] integrate ontology and natural language processing techniques for query answering.

Some of the existing endeavors on entity co-reference detection and resolution services [32, 30, 12] are also related to our work as the join operation in ALOQUS is made possible by the detection of co-referent URIs.

## 7 Conclusion and Future Work

In this paper, we proposed a novel approach that allows querying of Linked Data without requiring that the user have intimate knowledge of individual datasets

and interconnecting relationships. The basic idea of our approach is to make use of ontology alignment systems for querying. Our system supports writing queries using just an upper level ontology (e.g., PROTON) or cross-domain ontologies (e.g., DBPedia) or any other ontology as the primary ontology for expressing queries. Our methodology allows automatic retrieval and merging of results for queries that involve resources indirectly linked in the LOD cloud. Using this approach, we are able to answer queries which cannot be answered by state of the art systems for LOD query processing. With our initial test harness and sample queries, we hope that our community will develop a resource for evaluating future efforts in alignment-based querying systems.

While the contributions in this paper provide a novel and, in our opinion, a scalable querying approach for LOD querying, there is also a lot of room for improvement. Given the fact that our method depends on one of the currently available alignment systems, ALOQUS has limitations that stem from the limitations of BLOOMS, our chosen alignment system. Present day alignment systems try to find direct mappings between two different concepts. However, there are cases where the two concepts might not align directly but only if there is a chain of mappings as exemplified in the R2R Framework.[30] Such mapping chains are currently not supported in ALOQUS. We believe that building a better alignment system is important and that alignment based querying systems like ALOQUS will greatly help users in writing queries without specifying exact relations and knowing datasets beforehand.

Future work includes incorporating query caching, analysis of query logs and optimization of query plans for faster query execution. We also aim to make use of VoID statistics [1] for source selection.

## References

1. Alexander, K., Hausenblas, M.: Describing linked datasets – on the design and usage of voiD, the vocabulary of interlinked datasets. In: In Linked Data on the Web Workshop (LDOW 09), in conjunction with 18th International World Wide Web Conference (2009)
2. Bishop, B., Kiryakov, A., Ognyanov, D., Peikov, I., Tashev, Z., Velkov, R.: Factforge: A fast track to the web of data. Semantic Web 2(2), 157–166 (2011)
3. Bizer, C., Heath, T., Berners-Lee, T.: Linked data – the story so far. International Journal of Semantic Web and Information Systems 5(3), 1–22 (2009)
4. Damova, M., Kiryakov, A., Simov, K., Petrov, S.: Mapping the central LOD ontologies to PROTON upper-level ontology. In: The Fifth International Workshop on Ontology Matching (2010)
5. Damova, M., Petrov, S., Simov, K.: Mapping data driven and upper level ontology. In: Dicheva, D., Dochev, D. (eds.) Artificial Intelligence: Methodology, Systems, and Applications, Lecture Notes in Computer Science, vol. 6304, pp. 269–270. Springer Berlin / Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-15431-7_31

[30] http://www4.wiwiss.fu-berlin.de/bizer/r2r/

6. David, J., Euzenat, J., Scharffe, F., dos Santos, C.T.: The Alignment API 4.0. Semantic Web 2(1), 3–10 (2011)
7. David, J., Guillet, F., Briand, H.: Matching directories and OWL ontologies with AROMA. In: Yu, P.S., Tsotras, V.J., Fox, E.A., Liu, B. (eds.) Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, November 6-11, 2006. pp. 830–831. ACM (2006)
8. Duan, S., Fokoue, A., Srinivas, K., Byrne, B.: A clustering-based approach to ontology alignment. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N.F., Blomqvist, E. (eds.) The Semantic Web – ISWC 2011 – 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I, Lecture Notes in Computer Science, vol. 7031, pp. 146–161. Springer, Heidelberg (2011)
9. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., dos Santos, C.T.: Ontology alignment evaluation initiative: Six years of experience. Journal on Data Semantics 15, 158–192 (2011)
10. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L.: Sweetening Ontologies with DOLCE. In: EKAW 2002. LNCS, vol. 2473, pp. 223–233 (2002)
11. Giunchiglia, F., Autayeu, A., Pane, J.: S-Match: An open source framework for matching lightweight ontologies. Semantic Web (2011), to appear. Available from http://www.semantic-web-journal.net/
12. Glaser, H., Jaffri, A., Millard, I.: Managing co-reference on the semantic web. In: WWW2009 Workshop: Linked Data on the Web (April 2009)
13. Hartig, O., Bizer, C., Freytag, J.C.: Executing SPARQL queries over the Web of Linked Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) The Semantic Web – ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings. LNCS, vol. 5823, pp. 293–309 (2009)
14. Hartig, O., Langegger, A.: A database perspective on consuming linked data on the web. Datenbank-Spektrum 10(2), 57–66 (2010)
15. Jain, P., Hitzler, P., Sheth, A.P., Verma, K., Yeh, P.Z.: Ontology alignment for linked open data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) The Semantic Web – ISWC 2010 – 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I. Lecture Notes in Computer Science, vol. 6496, pp. 402–417. Springer, Heidelberg (2010)
16. Jain, P., Hitzler, P., Yeh, P.Z., Verma, K., Sheth, A.P.: Linked Data is Merely More Data. AAAI Spring Symposium "Linked Data Meets Artificial Intelligence" (2010)
17. Jain, P., Yeh, P.Z., Verma, K., Vasquez, R.G., Damova, M., Hitzler, P., Sheth, A.P.: Contextual ontology alignment of LOD with an upper ontology: A case study with Proton. In: Antoniou, G., Grobelnik, M., Simperl, E.P.B., Parsia, B., Plexousakis, D., Leenheer, P.D., Pan, J.Z. (eds.) The Semantic Web: Research and Applications – 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29-June 2, 2011, Proceedings, Part I. Lecture Notes in Computer Science, vol. 6643, pp. 80–92. Springer, Heidelberg (2011)
18. Langegger, A., Wöß, W.: SemWIQ - Semantic Web Integrator and Query Engine. In: Hegering, H.G., Lehmann, A., Ohlbach, H.J., Scheideler, C. (eds.) INFORMATIK 2008, Beherrschbare Systeme – dank Informatik, Band 2, Beiträge der 38. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 8.-13. September, in München. LNI, vol. 134, pp. 718–722. GI (2008)

19. Li, J., Tang, J., Li, Y., Luo, Q.: RiMOM: A dynamic multistrategy ontology alignment framework. IEEE Transactions on Knowledge and Data Engineering 21(8), 1218–1232 (2009)
20. Lopez, V., Fernandez, M., Motta, E., Stieler, N.: PowerAqua: supporting users in querying and exploring the Semantic Web content. Semantic Web journal (2012), to appear. Available from http://www.semantic-web-journal.net/
21. Mascardi, V., Cord, V., Rosso, P.: A comparison of upper ontologies. Tech. rep., Dipartimento di Informatica e Scienze dell'Informazione (DISI), Università degli Studi di Genova (2006)
22. de Melo, G., Suchanek, F., Pease, A.: Integrating YAGO into the Suggested Upper Merged Ontology. In: 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2008), November 3-5, 2008, Dayton, Ohio, USA, Volume 1. IEEE Computer Society (2008)
23. Mena, E., Illarramendi, A., Kashyap, V., Sheth, A.P.: Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. Distributed and Parallel Databases 8(2), 223–271 (2000)
24. Niles, I., Pease, A.: Towards a Standard Upper Ontology. In: Proceedings of the International Conference on Formal Ontology in Information Systems – Volume 2001. pp. 2–9. ACM, New York (2001)
25. Oberle, D., et al.: DOLCE ergo SUMO: On Foundational and Domain Models in the SmartWeb Integrated Ontology (SWIntO). Journal of Web Semantics 5(3), 156–174 (2007)
26. Prud'hommeaux, E., Seaborne, A.a. (eds.): SPARQL query language for RDF. W3C Recommendation (15 January 2008), available at http://www.w3.org/TR/rdf-sparql-query/
27. Quilitz, B., Leser, U.: Querying Distributed RDF Data Sources with SPARQL. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings. Lecture Notes in Computer Science, vol. 5021, pp. 524–538. Springer, Heidelberg (2008)
28. Reed, S., Lenat, D.: Mapping Ontologies into Cyc. Tech. rep., Cycorp, Inc (2002), available from http://www.cyc.com/doc/white_papers/
29. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: Optimization techniques for federated query processing on linked data. In: Antoniou, G., Grobelnik, M., Simperl, E.P.B., Parsia, B., Plexousakis, D., Leenheer, P.D., Pan, J.Z. (eds.) The Semanic Web: Research and Applications – 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29 – June 2, 2011, Proceedings, Part II. Lecture Notes in Computer Science, vol. 6644, pp. 601–616. Springer, Heidelberg (2011)
30. Song, D., Heflin, J.: Domain-independent entity coreference in RDF graphs. In: Proceedings of the 19th ACM international conference on Information and knowledge management, CIKM'10. pp. 1821–1824. ACM, New York (2010)
31. Terziev, I., Kiryakov, A., Manov, D.: Base Upper-level Ontology (BULO) guidance. Deliverable 1.8.1, SEKT project (July 2005)
32. Tummarello, G., Delbru, R.: Entity coreference resolution services in Sindice.com: Identification on the current web of data. In: Bouquet, P., Halpin, H., Stoermer, H., Tummarello, G. (eds.) Proceedings of the 1st IRSW2008 International Workshop on Identity and Reference on the Semantic Web, Tenerife, Spain, June 2, 2008. CEUR Workshop Proceedings, vol. 422. CEUR-WS.org (2008)