

Alignment-based approximate SPARQL querying on linked open data

Yu Liu^{1, 2}, Lei Chen^{1*}, Shihong Chen¹

¹School of Computer Science and Technology, Wuhan University, Wuhan 430072, Hubei, China

²School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan 430065, China

Received 1 June 2014, www.cmnt.lv

Abstract

With the growth of Linked Open Data, more and more applications are developed to take full advantage of its massive data. However, all these applications face an inevitable problem - how to retrieve information from these datasets with different schemas, which results in that a query for a dataset may get none answer from other datasets. To solve this problem, ontology alignment has been adopted in some Linked Open Data querying systems. In this paper, we follow this idea and make further efforts to find more approximate answers by employing relations and probability values in the result of ontology alignment. The fundamental of our method is the similarity between entities, which is used to evaluate the similarity of rewritten query relative to original query. In order to facilitate user to query other dataset with original query, an algorithm for alignment-based approximate querying is proposed. In experiments, the SPARQL queries for DBpedia are rewritten on the basis of alignment result between DBpedia and YAGO. The results of experiments show that alignment-based approximate querying can not only retrieve approximate results, but also overcome the problem caused by imprecise result of ontology alignment, which is very common for most of alignment techniques.

Keywords: Linked Open Data, Approximate Querying, Ontology Alignment, SPARQL

1 Introduction

Since the W3C Linked Open Data (LOD) project launched in 2007, more and more datasets were published in the web of Data, which have covered a diversity of areas such as nature, geography, government, and so on. It offered a great incentive for researchers to develop various applications based on the LOD. The Traffic LarKC combined DBpedia, a central part of LOD, with an eventful wrapper and datasets of two Milano municipalities to implement an intelligent question answering system about the traffic [1]. On the basis of information theory, Meymandpour R. et al. took advantage of LOD in the domain of university to rank universities and compare the results with the international ranking system [2]. Obviously, the inevitable problem for those applications is how to retrieve information from the LOD efficiently.

There are several approaches to query the LOD and each of them has its own advantages and limitations [3]. Data warehouses and query federation, two traditional approaches for query answering over distributed data, can be employed to execute the complex and structured queries over LOD [4, 5]. The search engine for the LOD can retrieve the web by following RDF links and provide query interfaces for users to search information in the LOD, just as Sindice [6]. In contrast to above methods, traversal based query execution over LOD, a novel query

execution paradigm, can intertwine query pattern matching over a continuously growing dataset with the traversal of links in order to discover data that might be relevant to answer the executed query [7]. However, no matter what approach the user adopts, the query patterns must be provided in advance, that means the user should be familiar with all schemas of the datasets he wants to query.

In fact, it is an impossible task for a user because the number of datasets in LOD increases continuously and each dataset always has a particular schema. For example, DBpedia [8] and YAGO [9] have their respective properties to express the relationships between the instances even if two properties have the same meaning, such as “geo:long” and “yago:hasLongitude” shown in Figure 1. Suppose a user want to query the information of longitude, latitude and population about the city Chengdu, the SPARQL query in formula (1)¹ can retrieve the results from DBpedia, but YAGO returns nothing because of the mismatches between query patterns and schemas of YAGO. Except for the above inconvenience, querying on a single dataset may lose the chance to get answer that exists in other datasets. In Figure 1, the property “dbo:capital” between “dbr:Sichuan” and “dbr:Chengdu” is not stated in DBpedia, so that the query pattern “dbr:Sichuan dbo:capital ?c” finds nothing in DBpedia. But, “yago:Sichuan yago:hasCapital ?c”, the query pattern with the same meaning, can make up for the

* Corresponding author e-mail: Chen_lei0605@sina.com

¹All queries in this paper omit the definition of prefixes declared in Figure 1

discount in DBpedia by obtaining the answer from YAGO.

```
Select ?lat ?long ?p
Where { dbr:Chengdu geo:lat ?lat.
        dbr:Chengdu geo:long ?long.
        dbr:Chengdu dbo:populationTotal ?p.}
```

In order to solve those problems, the approach that user can follow is ontology alignment. By combining the mechanism of query federation with the results of ontology alignment, an alignment based Linked Open Data querying system (ALOQUS) was proposed in [10]. It can map concepts in an upper ontology or domain specific ontology to concepts in other datasets, providing the capability to answer queries which cannot be answered by other state of the art system for LOD query processing. In this paper, we follow the idea of ALOQUA and make further efforts. Our method can find more approximate answers and overcome the problem caused by imprecise result of ontology alignment. The contribution of this paper is three-fold:

- (1) We measure the similarity of rewritten query relative to the original query. The calculation of similarity depends on the result of ontology alignment, which include not only the relations between entities, but also the probability values of those relations.
- (2) We propose an algorithm for alignment-based approximate querying, which can help user to query other dataset by using the original query and result of ontology alignment.
- (3) We prove the validity of our approach through experiments on real-world datasets of LOD. The results show that alignment-based approximate querying can retrieve more answers.

The remainder of this paper is organized as follows. Section 2 discusses related work. In section 3, the

SPARQL query rewriting is described with a formal way. How to calculate similarity between entities and queries on the basis of alignment result are explained in section 4. Section 5 proposes an algorithm for alignment-based approximate querying. The experiment results and their evaluations are presented in section 6. The conclusion of this paper is in section 7.

2 Related work

Hurtado et al. propose an RDF query relaxation method through RDF(s) entailment producing more general queries for retrieving potential relevant answers [11]. In order to ensure the desired cardinality and quality of answers, Huang et al. use the similarity between relaxed queries and original query to control the relaxation process [12, 13]. However, these works relax the original query based on a single schema, which is not suitable for various schemas of datasets in LOD. In [14, 15], Reddy et al. attempt to make use of ontologies available on the web of data to produce approximate answers, and integrate the approximate steps with query execution to improve the performance of query processing. Nevertheless, the similarity measure of relaxed SPARQL queries and the experiments in [14] are still based on the dataset with a single schema.

The purpose of ontology alignment is to find the sets of correspondences between entities belonging to the matched ontologies [16]. The result of ontology alignment is generated by hand or by ontology matchers and can be used for merging ontologies, linking datasets and transforming queries [17]. Just like ALOQUS introduced in [10], the best alignments in the result of BLOOMS mapping are used to transform the sub-queries, which are executed in the corresponding end-points later.

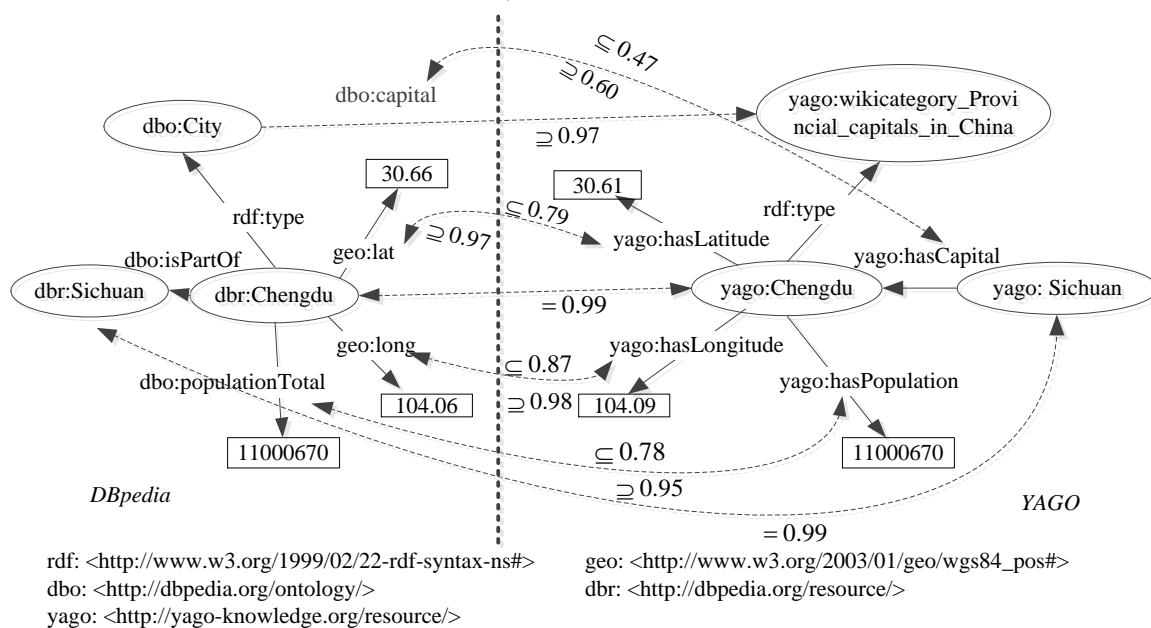


FIGURE 1 An example of ontology alignment between DBpedia and YAGO

Here, BLOOMS is an ontology matcher for schema alignment and can find the relations of including and equivalence between the classes belonging to the matched ontologies [18]. In general, relations found by an ontology matcher have some associated metadata. A frequently-used metadata element is confidence in the correspondence (typically in the [0, 1] range). The higher confidence means the higher likelihood that the relation holds [19]. Some ontology matchers use probability to describe the likelihood of relation between the entities, such as probability scores in PARIS (Probabilistic Alignment of Relations, Instances, and Schema) [20]. PARIS is a holistic ontology matcher--aligning not only instances but also properties and classes, which is a significant feature that can be employed to transform queries. Figure 1 presents some alignment results of PARIS about properties and instances between DBpedia and YAGO (e.g. “geo:lat” is a sub relation of “yago:hasLatitude” with the probability score 0.79 (Although values presented in this paper only reserve two decimal fractions, the type of values is double in the implementation procedure of alignment-based approximate SPARQL query)). In this paper, we will take the most advantage of PARIS to implement the approximate SPARQL querying on LOD.

3 SPARQL query rewriting

With the purpose of retrieving the data stored in RDF format, W3C proposed SPARQL. The fundamental component of SPARQL query is triple pattern, which can be expressed as $(s, p, o) \in (I \cup V) \times (I \cup V) \times (I \cup V \cup L)$. Here, V is a set of variables, I is a set of IRIs (Internationalized URIs), and L is a set of literals. In general, a SPARQL query is built on some triple patterns with series operations of conjunction and disjunction. It is worth noting that other components of SPARQL query are not mentioned here. It is because that they have little influence on the SPARQL query rewriting.

Given Q is a SPARQL query and all IRIs in triple patterns of Q come from the dataset $D1$, Q can be denoted as Q^{D1} , which means the Q is designed for $D1$ and may retrieve some results from $D1$. Because of the diversity of datasets in LOD, the execution of Q^{D1} on another dataset $D2$ may return nothing, just as the example introduced above. By replacing IRIs in triple patterns of Q^{D1} with IRIs in $D2$, the SPARQL query rewriting can generate some new queries (denoted as Q^{D2}) that have the same or similar meaning with Q^{D1} . Then, users do not have to know the schema of $D2$ to query it with Q^{D2} . For example, the query for DBpedia in formula (1) can be transformed to the query in formula (2), so that user can achieve the desired answers from YAGO. Despite having the significant advantage, the SPARQL query rewriting faces a vital problem--how to choose IRIs in $D2$ to replace the IRIs in Q^{D1} . Apparently, the high similarity between them is preferred.

```

Select  ?lat    ?long  ?p
Where { yago:Chengdu yago:hasLatitude ?lat.
        yago:Chengdu yago:hasLongitude ?long.
        yago:Chengdu yago:hasPopulation ?p.}
    
```

(2)

4 Alignment-based similarity

4.1 SIMILARITY BETWEEN ENTITIES

To compute the similarity of IRIs from two different datasets, the results of ontology alignment can be employed. The result of ontology alignment between $D1$ and $D2$ is a finite set S , including a certain number of 4-tuples. Formally, a 4-tuple can be written as $\langle e1, e2, r, p \rangle$. Here, $e1$ and $e2$ are entities in $D1$ and $D2$ respectively, and r is the relation between $e1$ and $e2$, such as equivalence (\equiv), including (\supseteq), included (\subseteq), and etc. The true probability value of the relation between $e1$ and $e2$ is p , which ranges from 0 to 1. For example, the relations between “geo:lat” and “yago:hasLatitude” in Figure 1 can be expressed as $\langle geo:lat, yago:hasLatitude, \supseteq, 0.97 \rangle$ and $\langle geo:lat, yago:hasLatitude, \subseteq, 0.79 \rangle$. Nevertheless, $\langle dbo:shipBeam, yago:hasLatitude, \subseteq, 0.78 \rangle$ also is a part of result when user adopts PARIS to align DBpedia and YAGO. At this time, the SPARQL query rewriting needs to decide which property is more similar with “yago:hasLatitude”.

Taking the diversity of 4-tuple into account, our method computes the similarity between entities according to different conditions. First, the relationship between instances is always equivalence, so similarity between $i1$ and $i2$ is p when S includes $\langle i1, i2, \equiv, p \rangle$. Here, $i1$ and $i2$ are instances in $D1$ and $D2$ respectively. In Figure 1, $\langle dbr:Chengdu, yago:Chengdu, \equiv, 0.99 \rangle$ implies that similarity between “dbr:Chengdu” and “yago:Chengdu” is 0.99, indicated as

$$Sim(dbr : Chengdu , yago : Chengdu) = 0.99 . \tag{3}$$

Second, class or property in dataset is usually viewed as a set that means r in 4-tuple could be any relations for set, which make the computation of similarity for them more complex than instance. For definiteness and without loss of generality, only computing methods for similarity between classes are introduced here. Given $c1$ and $c2$ are classes in $D1$ and $D2$ respectively, the similarity between two classes can be defined as:

$$Sim(c1,c2) = \frac{|c1 \cap c2|}{|c1 \cup c2|}, \tag{4}$$

where $|c1 \cup c2|$ is the cardinality of union of $c1$ and $c2$. It is clear that similarity between classes is in the [0, 1]. Suppose that the relation set of classes includes equivalence (\equiv), including (\supseteq) and included (\subseteq). Then, the calculations of similarity based on S are divided into the following cases:

(1) $\langle c1, c2, \equiv, p \rangle \in S$. In this case, similarity between $c1$ and $c2$ is p .

Proof: Given U includes all instances of classes in $D1$ and $D2$, and x is an element in U , then

$$P(c1 \equiv c2) = \frac{P(x \in c1 \cap c2 / x \in c1 \cup c2)}{P(x \in c1 \cup c2)} = \frac{|c1 \cap c2|}{|c1 \cup c2|} = \frac{|c1 \cap c2|}{|U|} = \frac{|c1 \cap c2|}{|c1 \cup c2|} = Sim(c1, c2) \quad (5)$$

(2) $\langle c1, c2, \supseteq, p1 \rangle \in S$ and $\langle c1, c2, \subseteq, p2 \rangle \in S$. In this case, similarity between $c1$ and $c2$ is

$$Sim(c1, c2) = \frac{p1 \cdot p2}{p1 + p2 - p1 \cdot p2} \quad (6)$$

Proof: Given x is an element in U , then

$$P(c1 \subseteq c2) = \frac{P(x \in c1 \cap c2 / x \in c1)}{|c1|} = \frac{|c1 \cap c2|}{|c1|} \quad (7)$$

$$P(c1 \supseteq c2) = \frac{P(x \in c1 \cap c2 / x \in c2)}{|c2|} = \frac{|c1 \cap c2|}{|c2|} \quad (8)$$

Because $P(c1 \subseteq c2) = p2$ and $P(c1 \supseteq c2) = p1$, then

$$|c1| = \frac{|c1 \cap c2|}{p2} \quad (9)$$

$$|c2| = \frac{|c1 \cap c2|}{p1} \quad (10)$$

Now, bring formula (9) and (10) into the definition of similarity, the conclusion can be achieved with the following inference.

$$Sim(c1, c2) = \frac{|c1 \cap c2|}{|c1 \cup c2|} = \frac{|c1 \cap c2|}{|c1| + |c2| - |c1 \cap c2|} = \frac{|c1 \cap c2|}{\frac{|c1 \cap c2|}{p2} + \frac{|c1 \cap c2|}{p1} - |c1 \cap c2|} = \frac{p1 \cdot p2}{p1 + p2 - p1 \cdot p2} \quad (11)$$

For example, $\langle dbo:School, yago:wordnet_school_108276720, \supseteq, 0.64 \rangle$ and $\langle dbo:School, yago:wordnet_school_108276720, \subseteq, 0.78 \rangle$ are in the result of ontology alignment when PARIS is adopted to align DBpedia and YAGO. Hence, the similarity between “ $dbo:School$ ” and “ $yago:wordnet_school_108276720$ ” is 0.54.

(3) $\langle c1, c2, \supseteq, p1 \rangle \in S$ or $\langle c1, c2, \subseteq, p2 \rangle \in S$. Since only one relation between two classes found by ontology matcher, formula (6) cannot be used directly in this case. For optimal performance, ontology matchers always set some thresholds to reduce the amount of computations,

such as PARIS assumes every value below to be zero [20]. It means the probability of undetected relation may be a value below the threshold. In addition, there are some relations between including probability and included probability. Table 1 presents some super classes of “ $dbo:School$ ” in YAGO, which are found by PARIS. Obviously, more general super classes of original class c is prone to have bigger value of included probability and leads to smaller value of including probability – and vice versa. In order to approximately simulate the above-mentioned relations between included probability and including probability, the formula (12) is adopted here.

TABLE 1 Some super classes of “ $dbo:Hospital$ ” in YAGO

DBpedia	Relation	YAGO	Probability
dbo:School	\subseteq	yago:wordnet_senior_high_school_108409617	0.47
dbo:School	\subseteq	yago:wordnet_secondary_school_108284481	0.61
dbo:School	\subseteq	yago:wordnet_school_108276720	0.78
dbo:School	\subseteq	yago:wordnet_institution_108053576	0.78
dbo:School	\subseteq	yago:wordnet_educational_institution_108276342	0.78
dbo:School	\subseteq	yago:yagoLegalActorGeo	0.79
dbo:School	\subseteq	yago:wordnet_entity_100001740	0.80

$$P(c \supseteq x) = \begin{cases} \frac{T \cdot (\max(c, \subseteq) - P(c \subseteq x))}{\max(c, \subseteq) - \min(c, \subseteq)} & \max(c, \subseteq) \neq \min(c, \subseteq) \\ T \cdot (1 - P(c \subseteq x)) & \max(c, \subseteq) = \min(c, \subseteq) \end{cases} \quad (12)$$

Here, T is threshold value of including relation, which is the upper limit for including relation, $\max(c, \subseteq)$ is the maximum probability value in the superclass set of c , $\min(c, \subseteq)$ is the minimum probability value in the superclass set of c . For example, $\max(dbo:School, \subseteq)$ is 0.80 and $\min(dbo:School, \subseteq)$ is 0.47, just as shown in Table 1. Once $P(c \supseteq x)$ gets calculated, the similarity between c and x can be easily achieved by bringing $P(c \supseteq x)$ and $P(c \subseteq x)$ into formula (6).

In fact, the computation of included probability based on including probability can adopt the same mechanism, just as formula (13) shows. In the same way, T' is threshold value of included relation; $\max(c, \supseteq)$ is the maximum probability value in the subclass set of c ; $\min(c, \supseteq)$ is the minimum probability value in the subclass set of c .

$$P(c \subseteq x) = \begin{cases} \frac{T' \cdot (\max(c, \supseteq) - P(c \supseteq x))}{\max(c, \supseteq) - \min(c, \supseteq)} & \max(c, \supseteq) \neq \min(c, \supseteq) \\ T' \cdot (1 - P(c \supseteq x)) & \max(c, \supseteq) = \min(c, \supseteq) \end{cases} \quad (13)$$

Although formula (12) and formula (13) are exactly similar, there are some tricks, that should be mentioned. For instance, T and T' are two key parameters, which

reflect the tendency of users for precision or recall. If T' is bigger than T , classes in subclass of c is prone to achieve higher similarity, that means these classes can take precedence to be selected during the process of query rewriting. Then, the precision of query results may increase according to the mechanism introduced in section 5.

4.2 SIMILAR ENTITIES RETRIEVAL

Based on the definition of similarity between entities, an algorithm is given here to obtain the similar entities of target entity. The steps of the algorithm consist of initialization, retrieval of similar entities, sorting the set of similar entities, and so on. The details of the algorithm are described as follows:

Algorithm SER

Input: an entity in original dataset e , a result set of ontology alignment $resultSet$, threshold value of including relation T , threshold value of included relation T' .

Output: a set of entity with similarity $similarSet$

```

1: Initialize  $similarSet$ 
2: find equivalent set of  $e$  from  $resultSet$ , denoted as  $equalSet$ 
3: find superclass set of  $e$  from  $resultSet$ , denoted as  $superSet$ 
4: find subclass set of  $e$  from  $resultSet$ , denoted as  $subSet$ 
5: for each  $x$  in  $equalSet$  do
6:   push  $\langle x, p \rangle$  into  $similarSet$  according to formula (5); Here,  $p$  is probability value, with which  $e$  equals with  $x$ .
7: end for
8: for each  $x$  in  $superSet$ 
9:   if  $x$  is not in  $similarSet$ 
10:    if  $x$  is in  $subSet$ 
11:      calculate the similarity  $s$  between  $x$  and  $e$  according to formula (6)
12:    else
13:      calculate the similarity  $s$  between  $x$  and  $e$  according to  $T$ , formula (12) and formula (6)
14:    end if
15:    push  $\langle x, s \rangle$  into  $similarSet$ 
16:  end if
17: end for
18: for each  $x$  in  $subSet$ 
19:   if  $x$  is not in  $similarSet$ 
20:     calculate the similarity  $s$  between  $x$  and  $e$  according to  $T'$ , formula (13) and formula (6)
21:     push  $\langle x, s \rangle$  into  $similarSet$ 
22:   end if
23: end for
24: sort  $similarSet$  in descending order by similarity
25: return  $similarSet$ 

```

4.3 SIMILARITY BETWEEN QUERIES

In general, a SPARQL query is composed of several triple patterns, so that the similarity between original triple pattern and rewritten triple pattern should be defined first. Suppose that $q^{D1}(s^{D1}, p^{D1}, o^{D1})$ is a triple pattern for the dataset $D1$ and $q^{D2}(s^{D2}, p^{D2}, o^{D2})$ is a rewritten triple pattern for the dataset $D2$, and then the similarity between q^{D1} and q^{D2} can be calculated by using formula (14).

$$Sim(q^{D1}, q^{D2}) = \frac{1}{3} (Sim(s^{D1}, s^{D2}) + Sim(p^{D1}, p^{D2}) + Sim(o^{D1}, o^{D2})) \quad (14)$$

Considering Figure 1 and Algorithm SER, $Sim(dbr:Chengdu, yago:Chengdu)$ is 0.99, $Sim(geo:lat, yago:hasLatitude)$ is 0.77, and then $Sim("dbr:Chengdu geo:lat ?c", "yago:Chengdu yago:hasLatitude ?c")$ is 0.92.

Given an original query $Q^{D1}(q_1^{D1}, q_2^{D1}, \dots, q_n^{D1})$ consists of n triple patterns and $Q^{D2}(q_1^{D2}, q_2^{D2}, \dots, q_n^{D2})$ is the rewritten query, then the similarity between Q^{D1} and Q^{D2} is

$$Sim(Q^{D1}, Q^{D2}) = \prod_{i=1}^n w_i \cdot Sim(q_i^{D1}, q_i^{D2}), \quad (15)$$

where $w_i \in (0,1]$ is the weight of triple pattern q_i^{D1} , which stands for the importance of q_i^{D1} in Q^{D1} . When the value of each w_i is set to 1, the similarity between formula (1) and formula (2) is 0.78.

5 Alignment-based approximate querying

On the basis of result of ontology alignment, alignment-based approximate querying can be implemented by executing the following procedures. At first, the query parser locates the entities in Q^{D1} that should be replaced later, and then puts them into a container for subsequent processing. It is obvious that variables and constants in Q^{D1} do not need to be added. Except for variables and constants, some IRIs that are widely used in most datasets should be skipped as well, such as "rdf:type", "owl:sameAs", and so on. In the second step, similar entities of each entity in the container are obtained by calling the algorithm SER. Then, some rewritten queries for the dataset $D2$ can be constructed by replacing original entities with similar entities. During the construction procedure, subject and object should be upside down when the property in dataset $D1$ is replaced by inverse property. For example, "dbo:creator" and "yago:created" have an inverse relationship. After sorting rewritten queries in descending order by similarity, the query engine executes these rewritten queries on dataset $D2$ one by one and records the query answers until the termination criterions are satisfied. In the following algorithm, the concrete termination criterions are not

given, because user can set particular criterions that fit the requirement of specific application. For some applications, it could be a reasonable termination criterion when top k of the most similar queries is executed on dataset D_2 . The details of algorithm for alignment-based approximate querying are described below.

Algorithm AAQ

Input: an query Q^{D_1} for the dataset D_1 , a result set of ontology alignment $resultSet$, threshold value of including relation T , threshold value of included relation T' .

Output: a result set with similarity $queryResult$

```

1: Initialize  $queryResult$ 
2: Initialize  $sContainer$ . Here,  $sContainer$  is a container
   that includes entities in  $Q^{D_1}$  and their similar entity in
   dataset  $D_2$ 
3: locate entities in  $Q^{D_1}$  that should be replaced
4: put these entities into a container, denoted as
 $eContainer$ 
5: for each  $e$  in  $eContainer$ 
6:   Initialize  $similarSet$ 
7:    $similarSet = SER(e, resultSet, T, T')$ 
8:   put  $\langle e, similarSet \rangle$  into  $sContainer$ 
9: end for
10:  $rqSet = RewrittenQuery(Q^{D_1}, sContainer)$ ; Here, the
    function  $RewrittenQuery$  is to construct similar
    queries;  $rqSet$  is a set that includes all rewritten query
    for dataset  $D_2$ 
11: sort  $rqSet$  in descending order by similarity
12: for each  $q$  in  $rqSet$ 
13:   if(terminationSatisfied())
14:     break
15:   end if
16:    $tmpResult = Engine.query(q, D_2)$ 
17:    $queryResult.add(tmpResult)$ 
18: end for
19: return  $queryResult$ 

```

By analysing the algorithm AAQ, it can be easily concluded that the size of rewritten query set is closely related to two factors--one is the number of entities that should be replaced; the other is the number of similar entities of each entity, which depends on the algorithm SER. Hence, the size of queryResult can be counted with Formula (16).

$$|queryResult| = \prod_{e \in eContainer} |SER(e, resultSet, T, T')|. \quad (16)$$

6 Experiments

To testify the availability of alignment-based approximate SPARQL query, two key datasets in LOD (DBpedia and YAGO) are chosen as testing data sets in this paper. The versions of datasets are the same as the

datasets used in [20]. The experiments show that the version of YAGO in [20] has no information about "rdf:type" facts of YAGO, therefore we download taxonomy data from the official website of YAGO and combine them with the above datasets. All data are imported into Virtuoso, which is an efficient RDF triple store and provides SPAQL query language support. Just as explained in section 2, the results of ontology alignment found by PARIS are employed to evaluate the similarity between entities and rewrite queries. In addition, T and T' are set to 0.3 and 0.1 respectively because they conform to the results of PARIS.

In experiments, we construct five SPARQL queries based on the schema of DBpedia, which are listed in Table 2. The purpose of these queries is to retrieve all instances of specified type and some attributes about them. For instance, No.1 query is to find population size and council area of settlements in DBpedia; No.2 query is to find latitude and longitude of schools in DBpedia. Table 3 presents some experimental results. Here, these queries for DBpedia produce plenty of similar queries for YAGO, such as No.3 query having 13675 rewritten queries, similarity of which range from 0.51 to 0.43. The main reason why No.3 query derives so many rewritten queries is that 13670 subclasses and 7 super classes of "dbo:Organisation" are found in the results of ontology alignment. It is not necessary to execute all rewritten queries on YAGO because some queries with low similarity may return lots of inaccurate results. In this paper, only top 10 of rewritten queries are executed on YAGO, and the last column of Table 3 shows the number of result, which do not take duplicate results into account. We also submit the original queries to DBpedia and count the number of result set respectively, which are listed in the column of "Result Number of Original Query on DBpedia". By comparing result numbers of original queries and rewritten queries, it can be concluded that rewritten queries can improve the recall to some extent.

The similarity and the result number about top 10 of rewritten queries are listed in Table 4, in which two characteristics should be noticed. The first one is that few queries in the front of list of rewritten queries have distinctive similarity. In the rewritten queries of No.4 query, the similarities of the first 3 rewritten queries are noticeably greater than others. That is because the entities replacing "dbo:Golfer" in these rewritten queries have both including and included relations with "dbo:Golfer" in the result of ontology alignment. "yago:wordnet_golfer_110136959", "yago:wikicategory_American_golfers", "yago:wikicategory_PGA_Tour_golfers" are these entities. Obviously, the rewritten queries with these entities are more likely to achieve greater similarity according to the computational method of similarity introduced in section 4. Second, the rewritten query with the greatest similarity may return nothing despite obtaining many results in most cases. In Table 4, No.3 query is a typical example. By executing the algorithm SER on the result of PARIS, it is found that the

similarity between “dbo:Organisation” and “yago:yagoLegalActor” is greater than the similarity between “yago:wordnet_organization_108008335” and “dbo:Organisation”, which results in “yago:yagoLegalActor” is used to replace “dbo:Organisation” in the top 1 of rewritten queries of No.3 query. However, for literal meaning and answer size “yago:wordnet_organization_108008335” is a better

substitute than “yago:yagoLegalActor”. The same phenomenon happens on No.5 query. The causation of this phenomenon is that the result of ontology alignment is not absolute precision. It means that approximate query is more suitable for the information retrieval based on ontology alignment--because the rewritten query with the “most” similar entity may miss chance to find the information meeting requirements.

TABLE 2 Some SPARQL queries based on the schema of DBpedia

No.	Queries fo DBpedia
1	SELECT ?s ?p ?c WHERE { ?s rdf:type dbo:Settlement. ?s dbo:populationTotal ?p. ?s dbo:councilArea ?c. }
2	SELECT ?s ?lat ?long WHERE { ?s rdf:type dbo:School. ?s geo:lat ?lat. ?s geo:long ?long. }
3	SELECT ?o ?c ?m WHERE { ?o rdf:type dbo:Organisation. ?c dbo:creator ?o. ?o dbo:motto ?m. }
4	SELECT ?g ?a WHERE { ?g rdf:type dbo:GolfPlayer. ?g dbo:award ?a. }
5	SELECT ?a ?n ?d WHERE { ?a rdf:type dbo:Artist . ?a dbo:notableWork ?n. ?a dbo:deathDate ?d }

TABLE 3 Some features of original queries and their rewritten query

No.	Number of Rewritten Query	Max of Similarity	Min of Similarity	Result Number of Original Query on DBpedia	Result Number of Top 10 of Rewritten Queries on YAGO
1	7500	0.52	0.41	495	210
2	2506	0.75	0.58	9304	2973
3	13675	0.51	0.43	5	23
4	187	0.7	0.5	754	29
5	8345	0.53	0.42	1310	93

TABLE 4 Experimental details about top 10 of rewritten queries

No	1	2	3	4	5					
Top10RQ.	Similarity	Result	Similarity	Result	Similarity	Result	Similarity	Result	Similarity	Result
1	0.52	108	0.75	2837	0.51	0	0.70	8	0.53	0
2	0.51	23	0.73	116	0.50	23	0.61	0	0.53	0
3	0.50	31	0.70	1	0.46	0	0.60	4	0.49	1
4	0.49	45	0.63	23	0.46	0	0.54	4	0.45	67
5	0.45	0	0.63	13	0.46	0	0.54	23	0.45	6
6	0.45	0	0.63	1	0.46	0	0.54	0	0.45	0
7	0.45	1	0.63	0	0.46	0	0.54	0	0.45	6
8	0.45	0	0.63	0	0.46	0	0.54	0	0.45	2
9	0.45	2	0.63	5	0.46	0	0.54	0	0.45	1
10	0.45	0	0.63	11	0.46	0	0.54	0	0.45	10

7 Conclusion

Aiming at the problem that various schemas of datasets make it inconvenience to retrieve information from LOD, an approach of approximate SPARQL querying based on ontology alignment is proposed in this paper. On the basis of the formal result of ontology alignment, our approach quantitatively measures the similarity between entities in different conditions, and then the similarity between rewritten queries. Further, our approach can use these rewritten queries to obtain approximate answers from other dataset. The experiments show that alignment-based approximate SPARQL querying can not only retrieve approximate answers, but also overcome the problem caused by imprecise result of ontology alignment, which is very common for the techniques of




ontology alignment. In the future work, we will improve alignment-based approximate querying from two aspects: one aspect is to increase the accuracy of ontology alignment between datasets in LOD; the other is to combine alignment-based approximate querying with other query execution paradigm, such as traversal based query execution over LOD [7].

Acknowledgments

This work was partly supported by the Science Guidance Project of Education Department of Hubei Province under Grant No. B2014085, Science Foundation for Yong Teachers of Wuhan University of Science and Technology under Grant No. 2012XZ015.

References

- [1] Celino I, Dell'Aglio D, Della E, et al. 2011 Integrating Machine Learning in a Semantic Web Platform for Traffic Forecasting and Routing *Proc. of the 3rd ESWC Workshop on Inductive Reasoning and Machine Learning*
- [2] Meymandpour R, Davis J G 2013 Ranking universities using linked open data *Proc. of the 6th Workshop on Linked Data on the Web*
- [3] Hartig O, Langegger A 2010 A database perspective on consuming linked data on the web *Datenbank-Spektrum* **10**(2) 57-66
- [4] Weiss C, Karras P, Bernstein A 2008 Hexastore: sextuple indexing for semantic web data management *Proc. of the 34th International Conference on Very Large Data Bases* **1**(1) 1008-19
- [5] Quilitz B, Leser U 2008 Querying distributed RDF data sources with SPARQL *Springer Berlin Heidelberg* 524-38
- [6] Oren E, Delbru R, Catasta M, et al. 2008 Sindice. com: a document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies* **3**(1) 37-52
- [7] Hartig O, Freytag J C 2012 Foundations of traversal based query execution over linked data *Proc. of the 23rd ACM conference on Hypertext and social media* 43-52
- [8] Bizer C, Lehmann J, Kobilarov G, et al. 2009 DBpedia-A crystallization point for the Web of Data *Web Semantics: Science, Services and Agents on the World Wide Web* **7**(3) 154-65
- [9] Hoffart J, Suchanek F M, Berberich K, et al. 2013 YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia *Journal of Artificial Intelligence* **194** 28-61
- [10] Joshi A K, Jain P, Hitzler P, et al. 2012 Alignment-based querying of linked open data *Proc. of On the Move to Meaningful Internet Systems* 807-24
- [11] Hurtado C A, Poulouvassilis A, Wood P T 2006 A Relaxed Approach to RDF Querying *Proc. of the 5th International Semantic Web Conference* 314-28
- [12] Hai Huang, Chengfei Liu, Xiaofang Zhou 2008 Computing Relaxed Answers on RDF Databases *Proc. of Workshop on Web Information Systems Engineering* 163-75
- [13] Hai Huang, Chengfei Liu, Xiaofang Zhou Approximating query answering on RDF databases *World Wide Web* **15** 89-114
- [14] Reddy B R K, Kumar P S 2010 Efficient approximate SPARQL querying of Web of Linked Data *Proc. of 6th international workshop on Uncertainty Reasoning for the Semantic Web* 37-48
- [15] Reddy K B R, Kumar P S 2013 Efficient trust-based approximate sparql querying of the web of linked data *Proc. of 9th international workshop on Uncertainty Reasoning for the Semantic Web* 315-30
- [16] Euzenat J, Shvaiko P 2007 Ontology matching *Springer Heidelberg*
- [17] David J, Euzenat J, Scharffe F, et al. 2011 The alignment api 4.0. *Journal of Semantic Web* **2**(1) 3-10
- [18] Jain P, Hitzler P, Sheth A P, et al. 2010 Ontology alignment for linked open data *Proc. of The 9th International Semantic Web Conference* 402-17
- [19] Shvaiko P, Euzenat J 2013 Ontology matching: state of the art and future challenges *IEEE Transactions on Knowledge and Data Engineering* **25**(1) 158-76
- [20] Suchanek F M, Abiteboul S, Senellart P 2011 Paris: Probabilistic alignment of relations, instances, and schema *Proc. of the 37th International Conference on Very Large Data Bases* **5**(3) 157-68

Authors	
	<p>Yu Liu, born on January 20, 1980, China</p> <p>Current position, grades: PhD candidate in School of Computer Science and Technology of Wuhan University. University studies: master degree in School of Computer Science and Technology of Huazhong University of Science and Technology. Scientific interests: semantic web and knowledge engineering.</p>
	<p>Lei Chen, born on November 21, 1978, China</p> <p>Current position, grades: instructor in School of Computer Science and Technology of Wuhan University. University studies: master degree in National University of Singapore and doctoral degree in Huazhong University of Science and Technology. Scientific interests: knowledge engineering and complex system.</p>
	<p>Shihong Chen, born on May 12, 1949, China</p> <p>Current position, grades: professor in School of Computer Science and Technology of Wuhan University. He also is the deputy director of the national engineering research center for multimedia software. Scientific interests: software engineering and knowledge engineering.</p>