UFOMQ: An Algorithm for Querying for Similar Individuals in Heterogeneous Ontologies

Yinuo Zhang[1(✉)], Anand Panangadan[2], and Viktor K. Prasanna[2]

[1] Department of Computer Science,
University of Southern California, Los Angeles, CA, USA
yinuozha@usc.edu
[2] Ming Hsieh Department of Electrical Engineering,
University of Southern California, Los Angeles, CA, USA
{anandvp,prasanna}@usc.edu

Abstract. The chief challenge in identifying similar individuals across multiple ontologies is the high computational cost of evaluating similarity between every pair of entities. We present an approach to querying for similar individuals across multiple ontologies that makes use of the correspondences discovered during ontology alignment in order to reduce this cost. The query algorithm is designed using the framework of fuzzy logic and extends fuzzy ontology alignment. The algorithm identifies entities that are related to the given entity directly from a single alignment link or by following multiple alignment links. We evaluate the approach using both publicly available ontologies and from an enterprise-scale dataset. Experiments show that it is possible to trade-off a small decrease in precision of the query results with a large savings in execution time.

1 Introduction

With the increasing use of ontologies for storing large amounts of heterogeneous data in enterprise-scale applications, there has been a corresponding interest in automatically discovering links between such ontologies [11]. Links between ontologies are represented as *alignments* between the entities contained in them. While a variety of approaches have been proposed in recent years to discover such alignments [3,11], much less work has been carried out in identifying similar individuals in different ontologies. The chief challenge in identifying similar individuals is the scale of the search space. Potentially, every individual represented in the ontologies has to be evaluated for its similarity to the query individual along with all of its properties. Such exhaustive evaluation of the search space is not feasible in enterprise-scale datasets.

We propose an approach to querying for similar individuals across multiple ontologies that makes use of the alignments discovered during the ontology linking process. Specifically, if the alignments represent the degree to which two entities in different ontologies share a particular type of relationship, then a query algorithm that returns individuals in decreasing order of their similarity to the target individual only needs to follow alignments starting from all of that

S. Madria and T. Hara (Eds.): DaWaK 2015, LNCS 9263, pp. 178–189, 2015.
DOI: 10.1007/978-3-319-22729-0_14

target individual's properties. Such a query mechanism can be designed using the framework of fuzzy logic. In prior work [15], we developed UFOM, a unified framework to generate ontology alignments for computing different types of correspondences. UFOM is based on a fuzzy representation of ontology matching. In this work, we present algorithms that utilize the fuzzy correspondences discovered by UFOM to efficiently query for all entities in an ontology that are similar to a given entity.

The algorithms can identify entities that are related to the given entity directly from a single alignment link (*direct matching*) or by following multiple alignment links (*indirect matching*). The algorithms are specified using a fuzzy extension of SPARQL (f-SPARQL [2]). The fuzzy SPARQL queries are then converted to crisp queries for execution. We evaluate this approach using both publicly available ontologies provided by the Ontology Alignment Evaluation Initiative (OAEI) campaigns and ontologies of an enterprise-scale dataset. Our experiments show that it is possible to trade-off precision of the similarity of the identified entities and the running time of the proposed query algorithms. Compared with a baseline approach (traversing all properties in an ontology), our proposed approach reduces execution time by 99 %.

The main contributions of this paper are (1) algorithms to efficiently identify similar entities in ontologies using links discovered during ontology alignment, (2) use of fuzzy SPARQL extensions to compactly represent similarity queries, and (3) quantitative evaluation of the approach on real-world datasets. The rest of the article is organized as follows. Section 2 gives an overview of the related work. In Sect. 3, we present background concepts and the problem definition. Section 4 summarizes our previous work on the UFOM ontology alignment approach. Section 5 describes how queries can be executed efficiently using the computed ontology alignment. In Sect. 6, we present the experimental evaluation of this query approach. We conclude in Sect. 7.

2 Related Work

Ontology matching is one of the key research topics in the Semantic Web community and has been widely investigated in recent years [9,11]. An ontology matching system discovers correspondences between entities in two ontologies. Most existing systems adopt multiple strategies such as terminological, structural, and instance-based approaches to fully utilize information contained in the ontologies.

However, exact matches do not always exist due to the heterogeneity of ontologies. Fuzzy set theory has recently been used for ontology matching in order to address this issue. Todorov et al. [14] propose an ontology matching framework for multiple domain ontologies. They represent each domain concept as a fuzzy set of reference concepts. The matches between domain concepts are derived based on their corresponding fuzzy sets. In [6], a rule base provided by domain experts is used in the matching process. In that system, both Jaccard coefficient and linguistic similarity are first calculated for each pair of entities. Then, the system uses the rule base to generate the final similarity measure of

each correspondence. The fuzzy set is used as a link between the preliminary similarities and the final one. Both of the above-described works adopt fuzzy set theory for the ontology matching task; however they do not provide a formal definition of a fuzzy representation of correspondence. Moreover, both systems are specific to equivalence type relations of instances. There is no generic framework to identify correspondences for different types of relations. [12] propose an approach for automatically aligning instances, relations and classes. They measure degree of matching based on probability estimate. They focus on discovering different relations between classes and relations. For instances, only equivalence relation is considered. [7] introduce a robust ontology alignment framework which can largely improve the efficiency of retrieving equivalent instances compared with [12]. In [15], we proposed a unified fuzzy ontology matching (UFOM) framework to address these two issues. Specifically, UFOM uses fuzzy set theory as a general framework for discovering different types of alignment across ontologies. It enables representation of multiple types of correspondence relations and characterization of the uncertainty in the correspondence discovery process. Section 4 summarizes the UFOM framework.

Federated ontologies facilitate the process of querying and searching for relevant information. Nowadays, hundreds of public SPARQL endpoints have been deployed on the web. However, most of these are works in progress in terms of discoverability, interoperability, efficiency, and availability [1]. Mora and Corcho [8] investigate the evaluation of ontology-based query rewriting systems. They provide a unified set of metrics as a starting point towards a systematic benchmarking process for such systems. [4] formalizes the completeness of semantic data sources. They describe different data sources using completeness statements expressed in RDF and then use them for efficient query answering. [10] presents a duplicate-aware method to querying over the Semantic Web. Specifically, they generate data summaries to reduce the number of queries sent to each endpoint. [13] proposes a query-specific matching method which generates path correspondences to facilitate query reformulation. All of these works aim to design a unified framework for query execution and information integration. However, none of them consider the uncertainty and ambiguity inherent in ontology correspondences. In our work, we present a query execution algorithm in which query execution is facilitated by using ontology correspondences with fuzzy relation types.

3 Problem Definition

The problem of ontology matching problem is to find an *alignment* A for a pair of ontologies O_1 and O_2. An ontology alignment is defined as follows [11],

Definition 1. *An* ***ontology alignment*** *A is a set of* correspondences *between entities of the matched ontologies O_1 and O_2.*

Definition 2. *An* ***ontology correspondence*** *is a 4-tuple: $< id, e_1, e_2, r >$, where id is the identifier for the given correspondence, e_1 and e_2 are the entities of the correspondence (e.g., properties in the ontology), and r is the relation type between e_1 and e_2 (e.g., equivalence and disjointedness).*

Note that in the above definition, the correspondence is exact, i.e., the relation r strictly holds between the ontology entities e_1 and e_2. However, in systems that have to automatically determine the set membership from real-world data, it is natural that a degree should be associated with the relation between the entities. The higher the degree is, the higher the likelihood that the relation holds between them. In order to represent the uncertainty in the correspondences, we presented a fuzzy variant of ontology alignment in [15].

Definition 3. *A **fuzzy ontology alignment** is a set of **fuzzy correspondences** in the form of 6-tuple:* $< id, e_1, e_2, r, s, c >$

where $s = \mu_r(e_1, e_2)$ is the *relation score* denoting the membership of (e_1, e_2) in relation r, and c is the *confidence score* computed while generating the correspondence. With this definition of fuzzy correspondence, we can extend the relation type set with other useful types such as *Relevance* [15].

A query execution is a process of retrieving a set of *individuals* $I = \{i_1, i_2, ..., i_n\}$ which are relevant to a given individual t. I and t can belong to either one ontology or multiple ontologies. In this paper, we study the case in which I and t come from two heterogeneous ontologies O_1 and O_2. The one-ontology case is a specialization of the multi-ontology one.

A brute force method is to compare t with all property values in O_2. The time complexity of this approach is $O(|O_2||E_2|)$ where $|O_2|$ is the number of individuals in O_2 and $|E_2|$ is the number of properties in O_2. The brute force approach is inefficient in terms of search time. In this paper, we describe how we can improve the query performance using fuzzy alignment information.

4 Unified Fuzzy Ontology Matching (UFOM)

The foundation of UFOMQ algorithm is the fuzzy ontology alignment derived using the unified fuzzy ontology matching (UFOM) framework [15]. UFOM computes fuzzy ontology correspondences based on both a *relation score* and a *confidence score* between every possible entity pair in the ontologies. In order to provide an extensible framework, every relation score (i.e., for every type of relation of interest) is computed from a set of pre-defined similarity functions. The framework is illustrated in Fig. 1.

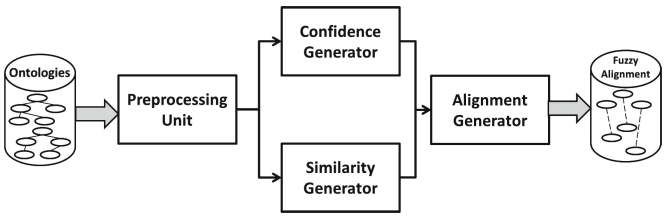

Fig. 1. Components of the UFOM system for computing fuzzy ontology alignment

UFOM takes two ontologies as input and outputs a fuzzy alignment between them. UFOM consists of four components: Preprocessing Unit (PU), Confidence Generator (CG), Similarity Generator (SG), and Alignment Generator (AG). PU classifies the entities in the ontologies based on their types. An entity can be either a class or a property. If it is a property, it is further identified as one of the following types: ObjectProperty, String DatatypeProperty, Datetime DatatypeProperty, and Numerical DatatypeProperty. Different score computation strategies are developed for different types of entities. CG computes a confidence score for each correspondence which reflects the sufficiency of the underlying data resources used to generate this correspondence. Volume and variety are the two major metrics considered in this step. SG generates a vector of similarities for each pair of entities. The similarities form the basis for computing correspondences with different types of relation. In UFOM, we consider four types of similarity: Name-based Similarity, Mutual Information Similarity, Containment Similarity, and Structural Similarity.

Name-based Similarity is calculated based on both the semantic similarity and syntactic similarity of between the names of the two properties. Mutual Information Similarity models the mutual information that exists between the *individuals* of one entity and the *domain* represented by the second entity. Containment Similarity models the average level of alignment between an instance of property e_1 and the most similar instance in property e_2. Structural Similarity represents the degree of structural similarity between two properties as they are represented within their ontologies. The detailed formulation is presented in [15]

The component AG is responsible for generating a set of fuzzy correspondences in the form of 6-tuples: $< id, e_1, e_2, r, s, c >$. It calculates the relation score r using a fuzzy membership function for each relation type. After both s and c are derived, AG prunes the correspondences with s and c less than predefined cutoff thresholds s_δ and c_δ. For instance, the following are examples of fuzzy correspondences.

$< 1, isbn : author, bkst : desc, relevance, 0.73, 0.92 >$
$< 2, isbn : author, bkst : desc, equivalence, 0.58, 0.92 >$
$< 3, isbn : author, bkst : pub, disjoint, 0.83, 0.75 >$

5 Query Execution

We now describe the UFOMQ algorithm to efficiently execute queries over two heterogeneous ontologies using pre-computed fuzzy ontology alignments. In order to take advantage of the fuzzy representation of the alignments, the query process consists of two phases: generating a fuzzy SPARQL query and then converting it to a crisp SPARQL query for execution. We adopt a specific fuzzy extension of SPARQL called f-SPARQL [2]. An example of f-SPARQL query is given below.

```
#top-k FQ# with 20
SELECT ?X ?Age ?Height WHERE{
   ?X rdf:type Student
```

```
    ?X ex:hasAge ?Age with 0.3.
    FILTER (?Age=not very young && ?Age=not very old) with 0.9.
    ?X ex:hasHeight ?Height with 0.7.
    FILTER (?Height close to 175 cm) with 0.8.
}
```

In this example, "not very young" and "not very old" are fuzzy terms, and "close to" is a fuzzy operator. Each condition is associated with a user-defined weight (e.g., 0.3 for age and 0.7 for height) and a threshold (e.g., 0.9 for age and 0.8 for height). The top 20 results are returned based on the score function [2].

1 Identify a set of properties $E_{direct} = \{e_1, e_2, ..., e_m\}$ in O_2 where e_j is in a fuzzy correspondence $< id, e^t, e_j, r, s, c >$ with $s \geq S$, $s \geq C$ and $r \in \{equivalence, relevance\}$, S and C are user-defined thresholds and e^t is t's identifier property;
2 Identify a set of property triples $E_{indirect} = \{\{e_1^1, e_1^2, e_1^3\}, ..., \{e_n^1, e_n^2, e_n^3\}\}$ from O_2 where e_j^1 is in a fuzzy correspondence $< id, e^t, e_j^1, r, s, c >$ with $s \geq S$, $s \geq C$ and $r \in \{equivalence, relevance\}$, and e_j^1 and e_j^2 are the properties of the same class (intermediate class) where e_j^2 is its identifier, and e_j^3 is the target property equivalent to e_j^2;
3 **for** *each* e_j *in* E_{direct} **do**
4 Generate a fuzzy SPARQL using the direct matching generation rule;
5 Calculate a seed vector $\overrightarrow{s}$=$\{s_{syn}, s_{sem}, s_{con}\}$ for each pair (t, v_x) where t is the given individual and v_x is a value in e_j;
6 Generate individuals with grades calculated by a relevance function of $\overrightarrow{s}$ in the instance ontology $onto_{ins}$;
7 **for** *each* $\{e_j^1, e_j^2, e_j^3\}$ *in* $E_{indirect}$ **do**
8 Generate a fuzzy SPARQL using the indirect matching generation rule;
9 Calculate a seed vector $\overrightarrow{s}$=$\{s_{syn}, s_{sem}, s_{con}\}$ for each pair (t, v_x) where t is the given individual and v_x is a value in e_j^1;
10 Generate individuals with grades calculated by a relevance function of $\overrightarrow{s}$ in the instance ontology $onto_{ins}$;
11 Generate a crisp SPARQL by computing the α-cut of the fuzzy terms based on the membership function and each graph pattern corresponds to a value in E_{direct} or $E_{indirect}$;
12 Return the individual set $I = \{i_1, i_2, ..., i_n\}$ by executing the crisp SPARQL over $onto_{ins}$;

Algorithm 1. UFOMQ - A query algorithm for UFOM

The UFOMQ algorithm is shown in Algorithm 1. The inputs to the algorithm are two ontologies O_1 and O_2, a set of fuzzy correspondences pre-computed using UFOM, and the target individual $t \in O_1$. The algorithm returns a set of individuals from O_2 that are similar to t. In our description, we state that the correspondences are either *equivalence* or *relevance*. However, the approach can

be extended to other types of relations provided the corresponding alignments are discovered by UFOM.

Steps 1 and 2 in Algorithm 1 identify related properties using fuzzy correspondences generated by UFOM. These properties are computed using two methods: *direct matching* and *indirect matching* (Fig. 2).

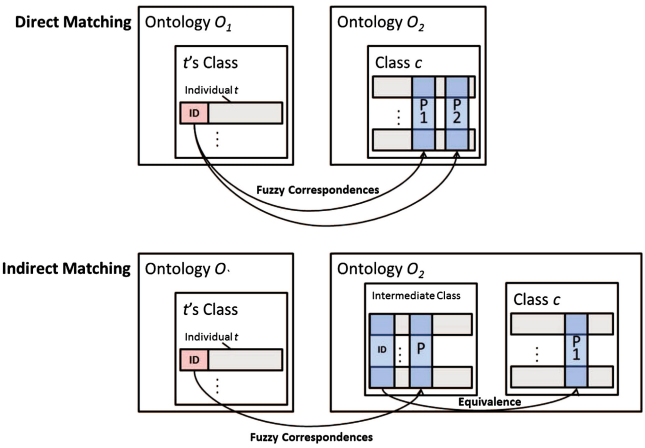

Fig. 2. Illustration of direct matching and indirect matching

For direct matching, we retrieve properties in O_2 which have fuzzy relations *equivalence* and *relevance*) with t's identifier property (e.g., *ID*) using the fuzzy alignment derived by UFOM. For example, the following SPARQL code retrieves only the *relevant* properties of *id* based on direct matching. Thresholds for relation score and confidence score (e.g., 0.5 and 0.7) are also specified in the query.

```
SELECT ?prop WHERE {
   ?prop ufom:type ''onto2_prop''.
   ?corr ufom:hasProp1 onto1:id.
   ?corr ufom:hasProp2 ?prop.
   ?corr ufom:relation ''Relevance''.
   ?corr ufom:score ?s.
   FILTER (?s > 0.5).
   ?corr ufom:conf ?c.
   FILTER (?c > 0.7).
}
```

Indirect matching is used to identify entities that do not share a single correspondence with t but are related via intermediate properties, i.e., more than one correspondence. We first identify all intermediate classes in O_2. The properties of such classes have a fuzzy relation with t's identifier property (e.g., *id*).

From these intermediate classes, we discover the properties which are equivalent to the identifier of the intermediate class. This equivalence relation is found by checking Object Properties in O_2. In contrast to direct matching which outputs a set of properties, indirect matching produces a collection of triples in the form of (e^1, e^2, e^3), where e^1 is the intermediate class' property with fuzzy relations with t's identifier property, e^2 is the intermediate class' identifier property, and e^3 is the target property equivalent to e^2. An example of the indirect matching approach for the relevance relation as expressed in SPARQL is shown below. *prop1*, *prop2*, and *prop3* correspond to e^1, e^2 and e^3 respectively.

```
SELECT ?prop1 ?prop2 ?prop3 WHERE {
   ?prop1 ufom:type ‘‘onto2_prop’’.
   ?prop1 rdfs:domain ?class.
   ?prop2 ufom:idof ?class.
   ?prop3 ufom:type ‘‘onto2_prop’’.
   ?prop3 rdfs:range ?class.
   ?corr ufom:hasProp1 onto1:id.
   ?corr ufom:hasProp2 ?prop1.
   ?corr ufom:relation ‘‘Relevance’’.
   ?corr ufom:score ?s.
   FILTER (?s > 0.5).
   ?corr ufom:conf ?c.
   FILTER (?c > 0.7).
}
```

Given the properties discovered by direct matching (*prop*) and indirect matching (*prop1*, *prop2* and *prop3*), we can build fuzzy SPARQL queries based on the rules expressed in Steps 4 and 8:

```
#top-k FQ# with 20
SELECT ?d WHERE {
   ?x onto2:id ?d.
   ?x onto2:prop ?p.
   FILTER (?p relevant to t) with 0.75.
}

#top-k FQ# with 20
SELECT ?d WHERE {
   ?x onto2:prop1 ?p.
   ?x onto2:prop2 ?c.
   ?y onto2:prop3 ?c.
   ?y onto2:id ?d.
   FILTER (?p relevant to t) with 0.75.
}
```

In the above rules, t is the given individual (the identifier used to represent the individual) and "relevant-to" is the fuzzy operator.

Since, eventually the fuzzy queries will have to be converted to crisp ones, we calculate a seed vector $\overrightarrow{s}=\{s_{syn}, s_{sem}, s_{con}\}$ for each value pair (t, v_x) where t is the given value (e.g., identifier of the given individual) and v_x is the value in the matched properties (e.g., "onto2:prop") (Steps 5 and 9). $\overrightarrow{s}$ represents multiple similarity metrics including syntactic, semantic, and containment similarities as described in [15]. The results are used to calculate the relevance scores which are stored as individuals in the instance ontology $onto_{ins}$ (Steps 6 and 10). In Step 11, we compute the α-cut of the fuzzy terms based on the membership function in order to convert to a crisp query. The resulting crisp SPARQL consists of multiple graph patterns and each of these corresponds to a matched property. The individual set I is derived by executing this crisp SPARQL query. An example of such a crisp SPARQL query returning individuals ranked based on their membership grades is shown below.

```
SELECT ?d WHERE {
   ?x onto2:id ?d.
   ?x onto2:prop ?p.
   ?ins onto_ins:value1 ?p.
   ?ins onto_ins:value2 t.
   ?ins onto_ins:type ''relevance''.
   ?ins onto_ins:grade ?g.
   FILTER (?g ≥ 0.75).
}
ORDER BY DESC(?g)
```

Using UFOMQ, the computation cost for retrieving relevant instances can be reduced. The time complexity of the UFOMQ algorithm is $O(|O_2||E_2(t)|)$ where $|E_2(t)|$ is the number of properties in O_2 which have fuzzy correspondences with t's identifier property. We evaluate the computation cost of UFOMQ on datasets in Sect. 6.

6 Experimental Evaluation

In this section, we present the results of applying the UFOMQ approach to two datasets. The first is publicly available ontologies from the Ontology Alignment Evaluation Initiative (OAEI) campaigns [5]. The second dataset comprises of ontologies of an enterprise-scale dataset.

OAEI Datasets. We first performed a set of experiments to evaluate the query execution process. The dataset is the Instance Matching (IM) ontology[1] from OAEI 2013. The dataset has 5 ontologies and 1744 instances. The fuzzy alignment is generated first using UFOM [15]. Then, we initialize 10 individuals from

[1] http://islab.di.unimi.it/im_oaei_2014/index.html.

one of the ontologies and retrieve related individuals from the other ontologies. WordNet[2] and DBPedia[3] are used to retrieve the synset and similar entities of a given individual. The membership grade threshold is set to 0.75. Figure 3 shows the performance of our query execution component on the IM ontology. Each data point is generated by averaging the results of 10 individuals.

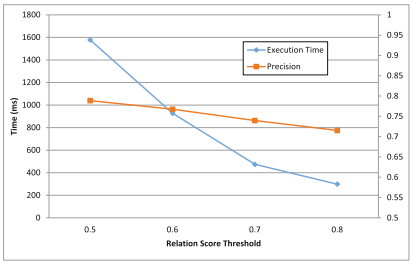

Fig. 3. Precision and execution time on applying UFOM query execution to the instance matching ontology

As the relation score threshold increases, both precision and running time for query execution decrease. This is because the number of correspondences decreases when we raise the relation score threshold. As a result, we have fewer correspondences to consider when we generate crisp queries and therefore the computational time is reduced. The reason precision also decreases is that we lose some true correspondences when we increase the relation score threshold. Those correspondences can help in finding more related individuals. However, as shown in Fig. 3, an increase in the threshold from 0.5 to 0.8 causes precision to decrease by only 9.3 % while execution time is reduced by 81.1 %. This indicates that the elimination of correspondences caused by increasing the threshold do not affect retrieving correct related individuals significantly. This is because the remaining correspondences are still sufficient to find connections between classes. In terms of querying for similar individuals, the correspondences between the same pair of classes have functional overlap.

Enterprise-Scale Dataset. For the evaluation of querying for related individuals, we considered two ontologies from an enterprise-scale information repository. Each ontology focuses on a different application area, but they are related in terms of the entities that they reference. Ontology O_1 has 125,865 triples. Ontology O_2 has 651,860 triples. Due to privacy concerns, we do not expose the real names of the properties and ontologies. We considered two classes, C_1 and C_2, in O_1 and two classes, C_3 an C_4, in O_2.

We identified 29 fuzzy correspondences between these two ontologies using UFOM. To evaluate query performance, we selected 10 representative individuals

[2] http://wordnet.princeton.edu/.
[3] http://dbpedia.org/.

Table 1. Query Execution Time (UFOM vs Baseline)

Scenario	UFOM(ms)	Baseline(ms)
C_1 to C_3	259	35974
C_2 to C_3	173	25706
C_1 to C_4	487	53937
C_2 to C_4	401	45752

from C_1 or C_2 and retrieve their relevant instances from C_3 or C_4 using the fuzzy alignment. Both precision and recall achieve 1.0 after we verified the results with the ground truth obtained by manually examining the ontologies for each of the automatically retrieved entities. We also generated the average execution time and the results are shown in Table 1. Compared with the baseline approach which traverses the values of all properties in O_2, our proposed approach reduces the execution by 99 % on average.

7 Conclusion

We presented the UFOMQ algorithm which enables scalable and efficient querying for related entities over heterogeneous ontologies. UFOMQ uses the fuzzy correspondences discovered during ontology alignment as computed by the previously developed UFOM framework. The query algorithm exploits the redundancy in the correspondences between classes — similar entities can be identified by following the strongest correspondences, not necessarily all the correspondences. In experiments performed on publicly available datasets, the query algorithm achieves a trade-off between precision of the returned query result and the computational cost of the query execution process. We also demonstrated the efficiency of UFOMQ in large enterprise-scale datasets.

For future work, we will develop query optimization strategies to facilitate efficient query execution. We will also adopt different entity identification techniques to improve the usability of the ontology alignment and query framework.

Acknowledgment. This work is supported by Chevron U.S.A. Inc. under the joint project, Center for Interactive Smart Oilfield Technologies (CiSoft), at the University of Southern California.

References

1. Buil-Aranda, C., Hogan, A., Umbrich, J., Vandenbussche, P.-Y.: SPARQL web-querying infrastructure: ready for action? In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 277–293. Springer, Heidelberg (2013)

2. Cheng, J., Ma, Z.M., Yan, L.: f-SPARQL: a flexible extension of SPARQL. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010, Part I. LNCS, vol. 6261, pp. 487–494. Springer, Heidelberg (2010)
3. Choi, N., Song, I.-Y., Han, H.: A survey on ontology mapping. SIGMOD Rec. **35**(3), 34–41 (2006)
4. Darari, F., Nutt, W., Pirrò, G., Razniewski, S.: Completeness statements about RDF data sources and their use for query answering. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 66–83. Springer, Heidelberg (2013)
5. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Trojahn, C.: Ontology alignment evaluation initiative: six years of experience. In: Spaccapietra, S. (ed.) Journal on Data Semantics XV. LNCS, vol. 6720, pp. 158–192. Springer, Heidelberg (2011)
6. Fernández, S., Velasco, J.R., Marsá-Maestre, I., López-Carmona, M.A.: Fuzzyalign - a fuzzy method for ontology alignment. In: KEOD, pp. 98–107 (2012)
7. Lee, S., Lee, J., Hwang, S.-W.: Fria: fast and robust instance alignment. In: 22nd International World Wide Web Conference, WWW 2013, Rio de Janeiro, Brazil, 13–17 May 2013, Companion Volume, pp. 175–176 (2013)
8. Mora, J., Corcho, O.: Towards a systematic benchmarking of ontology-based query rewriting systems. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 376–391. Springer, Heidelberg (2013)
9. Rahm, E.: Towards large-scale schema and ontology matching. In: Bellahsene, Z., Bonifati, A., Rahm, E. (eds.) Schema Matching and Mapping. Data-Centric Systems and Applications, pp. 3–27. Springer, Heidelberg (2011)
10. Saleem, M., Ngonga Ngomo, A.-C., Xavier Parreira, J., Deus, H.F., Hauswirth, M.: DAW: Duplicate-AWare federated query processing over the web of data. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 574–590. Springer, Heidelberg (2013)
11. Shvaiko, P., Euzenat, J.: Ontology matching: state of the art and future challenges. IEEE Trans. Knowl. Data Eng. **25**(1), 158–176 (2013)
12. Suchanek, F.M., Abiteboul, S., Senellart, P.: PARIS: probabilistic alignment of relations, instances, and schema. PVLDB **5**(3), 157–168 (2011)
13. Tian, A., Sequeda, J.F., Miranker, D.P.: QODI: query as context in automatic data integration. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 624–639. Springer, Heidelberg (2013)
14. Todorov, K., Geibel, P., Hudelot, C.: A framework for a fuzzy matching between multiple domain ontologies. In: König, A., Dengel, A., Hinkelmann, K., Kise, K., Howlett, R.J., Jain, L.C. (eds.) KES 2011, Part I. LNCS, vol. 6881, pp. 538–547. Springer, Heidelberg (2011)
15. Zhang, Y., Panangadan, A., Prasanna, V.K.: Ufom: unified fuzzy ontology matching. In: IRI 2014 - Proceedings of the 15th International Conference on Information Reuse and Integration, San Francisco, CA, USA, 13–15 August 2014