A Semantic Web Repository for Managing and Querying Aligned Knowledge

James P. McGlothlin, Latifur Khan

The University of Texas at Dallas
Richardson, TX USA
{jpmcglothlin, lkhan}@ utdallas.edu

Abstract. Ontology alignment is the task of matching concepts and terminology from multiple ontologies. Ontology alignment is especially relevant in the semantic web domain as RDF documents and OWL ontologies are quite heterogenous, yet often describe related concepts. The end goal for ontology matching is to allow the knowledge sets to interoperate. To achieve this goal, it is necessary for queries to return results that include knowledge, and inferred knowledge, from multiple datasets and terminologies, using the alignment information. Furthermore, ontology alignment is not an exact science, and concept matchings often involve uncertainty. The goal of this paper is to provide a semantic web repository that supports applying alignments to the dataset and reasoning with alignments. Our goal is to provide high performance queries that return results that include inference across alignment matchings, and rank results using certainty information. Our semantic web repository uses distributed inference and probabilistic reasoning to allow datasets to be efficiently updated with ontology alignments. We materialize the inferred, aligned data and make it available in efficient queries.

1 Introduction

We shall begin this paper by defining the high level problem we wish to solve and the system goals. Given multiple datasets (RDF documents) and ontologies, the goal is to allow queries against the complete knowledge set. The queries should be able to be specified using the terminologies from any of the ontologies, or from a new common global terminology. The queries should return all relevant knowledge, including inferred triples and triples that are specified using different, but corresponding, terminologies. The queries should rank results based on confidence values, and should enable selection based on probability conditionals. Furthermore, these queries should return such knowledge in a very timely manner.

Determining the alignments or matchings is the obvious first task. If two datasets use different terminology, then the concepts must be aligned to enable queries across both knowledge sets. Ontology alignment is already a well-researched area. Our contribution is to allow these alignments to be applied to the dataset, to transform the data accordingly, and to allow queries against the aligned results.

In [2], we first introduced RDFKB (Resource Description Framework Knowledge Base), a bit vector schema that is uniquely able to materialize inferred triples without a performance penalty. In [6], we introduced a multiple bit vector schema using thresholds to support efficient queries involving probability. RDFKB is the only semantic web repository to materialize uncertain information and to support queries involving probabilistic inference.

Our solution in this paper builds on these technologies. We propose to materialize all inferred triples including those inferred by reasoning with alignment matchings. In fact, we propose to implement all alignment matches as inference rules. We also propose to use the threshold vector schema as our solution for propagating and querying similarity measurements.

However, this paper involves more than just utilizing previous versions of RDFKB to apply alignments. We must be able to support distributed inference across multiple datasets, which changes our data management schema. We must be able to add inference rules to an existing dataset, which changes our inference rules specifications. Ontology alignment is a fluid process. Similarity measures can fluctuate through user feedback, additional machine learning, etc. Therefore, we must be able to change or even delete inference rules from the system. Also, many times the alignment goal is to transform the data into new terminology, thus replacing the original data. Finally, trust factors should be able to be associated with data origins, in order to facilitate handling conflicting information during dataset merges.

In Section 2, we will briefly overview the RDFKB schemata and our previous work. In Section 3, we will specify the new features that enable us to apply alignments. In Section 4, we will present experimental results, and, in Section 5, we make some conclusions and define some future work.

2 Background

RDFKB uses two schemata, one for data management, and one for queries. The data management schema is centered around a Triples table. This allows us to associate additional information (probability) with each triple, to optimize the addition and deletion processes, and to encapsulate the query schema from the user.

Our query schema includes two bit vector tables: POTable and PSTable. POTable contains 5 columns: the property, object, subjectbitvector, bitcount and threshold. The subject bit vector has a 1 for each subject that appears in a triple with the corresponding property and object and with probability >= threshold. PSTable contains 5 columns: property, subject, objectbitvector, bitcount and threshold. Bit vectors allow us to access entire collections of triples by reading a single tuple. Furthermore, joins and unions can be performed using efficient bit operations.

All inferred triples are materialized during addition time using forward chaining. [2] details how our bit vector schema is able to support inference materialization without a performance penalty, and describes our implementation of OWL inference rules. [6] details our probability solution using multiple bit vectors and thresholds, and demonstrates that adding probability does not reduce query performance.

3 New Features

Inference Rules. RDFKB uses inference rules to materialize inferred data. The actual instantiated inference rules are registered with RDFKB rather than implemented by RDFKB. At a high level, an inference rule defines that given a set of 1 or more triples ($A_1,\ldots,A_n$), we can conclude an additional triple B. For example, given <Professor1 type AssistantProfessor>, we can conclude <Professor1 type

Professor>. Using our high level definition above, it is obvious we can use inference rules to perform any alignment transformation.

The inference rule must define two methods *Infer()* and *Infer(triple T)*. *Infer(triple T)* returns the set of inferred triples that this inference rule can infer if T is added to the dataset. *Infer()* returns all triples that can be inferred across an entire dataset. This new method allows an inference rule to be added to an existing dataset, one of our requirements to support applying ontology alignments.

Provenance. In [6], RDFKB uses an InferenceCount value and forward chaining to enable updates and deletes. However, this will not allow us to update or delete an inference rule, which is required to support changing alignments and similarity measures. Therefore, we replace the InferenceCount field in the Triples table with Lineage, a foreign key into one of our provenance tables. The provenance tables are Users, Datasets, Events, and Dependencies. The Events table defines what inference event materialized an inferred triple, including the Inference Rule. This allows us query all triples materialized by an inference rule, so we are able to delete inference rules or update probabilities associated with inference rules.

The provenance tables also solve several of our other requirements. The Datasets table and Users table provide our trust factors. A concrete triple specifies its Dataset in the lineage column and we have replaced the Triples table with a collection of Triples tables. The Datasets table allows us to traverse over these tables, and inference rules can now query the entire collection of triples tables during the forward chaining process. This enables distributed inference, a core requirement for reasoning with alignment. We can apply any inference rule across multiple datasets and ontologies, which enables all possible alignment updates.

Remove(). Often, the goal is to transform the dataset to a new terminology rather than to just enable queries using the new terminology. The difference between these two scenarios is whether the original instances, using the original terminology, persist in the dataset. Inference rules can transform the dataset by instantiating inferred triples using the new terminology. To support deleting the original triples, we add a function *Remove(triple T)*. Remove, unlike *delete(triple T)* , removes the concrete triple without removing the inferred (transformed) triples.

4 Experimental Results

In [6], we show that we are faster than all existing semantic web repositories using all 26 queries defined by LUBM[3] and Barton Dataset[4]. In fact, we are faster than the next fastest solution, RDF-3X[1], by almost 3x times. There is no standard dataset and set of alignments for us to test our new features with. Our claim therefore is that since we are faster than other repositories for basic queries, and the current technologies for applying alignments (bridges, transformation APIs, etc) are all query time functions, we will also be much faster for queries involving alignment.

We have defined some simple experiments to demonstrate that we can apply alignment matchings to the dataset, and that queries are still efficient. For these experiments, we continued to use LUBM (with 67 million triples) and Barton dataset. Certain relationships between these datasets are natural since university students and

professors commonly author publications. Our tests utilize only the most common alignment matchings: sameAs and subClassOf. However, these tests verify that each of the requirements in Section 1 is satisfied. Performance times are listed in seconds.

The first experiment we performed was simply to align type Person by applying a sameAs inference rule matching the two Person classes. We applied this rule to the two datasets in 0.91s. We were then able to query *?s type Person* in 0.08s.

In the second experiment, we aligned instances. We choose 100,000 specific professors from LUBM and arbitrarily aligned them with 100,000 authors from Barton Dataset in 10.8s. We then queried all members of University0 who wrote a item of type text (0.03s), all professors who wrote a conference publication (0.24s), and all conferences which published a work by a professor (0.13s).

The third experiment we performed was to align the LUBM type Publication. We first aligned this with Item. This alignment took 0.91s. We queried all Items (0.07s) and Publications (0.08s). We then deleted this alignment and added a new alignment between Publication and Text (1.27s to delete and add the alignments). We changed the probability on this alignment from 1.0 to 0.97 (0.38s). We then queried and ranked items of type text(0.06s). The results from the Barton Dataset were first (p=1.0) followed by all items of type Publication from LUBM (p=0.97).

Finally, we used the subject types in Barton to align with the publicationResearch in LUBM. We took this a step further and used this information to guess the subjects associated with research groups and departments in LUBM. For example, if 6/7 publications written by members of a research group were on the topic of data mining, we concluded the research group related to data mining with p= 0.86. The main point in this experiment was to validate our ability to adjust alignments on the fly. For example, matching four researchers with authors increases the number of known topics for a research group and alters the similarity measure of the research group.

5 Conclusion and Future Work

We have defined a set of specific requirements necessary to allow ontology alignment to be applied to data in a semantic web repository. No existing semantic web repositories provide these features. We have presented experiments validating that we do provide these features, and we have presented performance numbers documenting the times required to apply various alignments and to query the results.

For future work, we would like to develop a complete benchmark and perform more elaborate experiments and comparisons. We also plan to use cloud computing solutions such as HBase and Hadoop[5] to further increase RDFKB's scalability.

6 References

1. Neumann, T., Weikum, G.: RDF-3X: a RISC-style engine for RDF. PVLDB(2008) 647-659
2. McGlothlin, J.P., Khan, L.R.: RDFKB: efficient support for RDF inference queries and knowledge management. In IDEAS(2009) 259-266
3. Lehigh University Benchmark (LUBM), http://swat.cse.lehigh.edu/projects/lubm
4. The Barton dataset, http://simile.mit.edu/wiki/Dataset:_Barton
5. Hadoop, http://hadoop.apache.org
6. Mcglothlin, J., Khan, L.: Materializing and Persisting Inferred and Uncertain Knowledge in RDF Datasets. In AAAI(2010)