UMBC_Ebiquity-SFQ: Schema Free Querying System

Zareen Syed[1], Lushan Han[1], Muhammad Rahman[1],
Tim Finin[1], James Kukla[2], Jeehye Yun[2]

[1]University of Maryland Baltimore County
1000 Hilltop Circle, MD, USA 21250
zsyed@umbc.edu, lushan1@umbc.edu, mrahman1@umbc.edu,
finin@cs.umbc.edu

[2]RedShred
5520 Research Park Drive, Suite 100
Baltimore, MD 21228
jkukla@redshred.net, jyun@redshred.net

Abstract. Users need better ways to explore large complex linked data resources. Using SPARQL requires not only mastering its syntax and semantics but also understanding the RDF data model, the ontology and URIs for entities of interest. Natural language question answering systems solve the problem, but these are still subjects of research. The Schema agnostic SPARQL queries task defined in SAQ-2015 challenge consists of schema-agnostic queries following the syntax of the SPARQL standard, where the syntax and semantics of operators are maintained, while users are free to choose words, phrases and entity names irrespective of the underlying schema or ontology. This combination of query skeleton with keywords helps to remove some of the ambiguity. We describe our framework for handling schema agnostic or schema free queries and discuss enhancements to handle the SAQ-2015 challenge queries. The key contributions are the robust methods that combine statistical association and semantic similarity to map user terms to the most appropriate classes and properties used in the underlying ontology and type inference for user input concepts based on concept linking.

Keywords: Information Storage and Retrieval, User Interfaces, Semantic web

1 Introduction

Developing interfaces to enable casual, non-expert users to query complex structured data has been the subject of much research over the past forty years. Since such interfaces allow users to freely query data without understanding its schema, knowing how to refer to objects, or mastering the appropriate formal query language, we call them as schema-free or schema-agnostic query interfaces. Schema-agnostic query interface systems address a fundamental problem in NLP, Database and AI: bridging the gap between a user's conceptual model of the world and the machine's representation.

adfa, p. 1, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Schema-agnostic query interface systems are challenged by three hard problems. First, we still lack practical interfaces. Unrestricted natural language interfaces (NLIs) are easy for people to use but hard for machines to process accurately. Today's NLP technology is still not reliable enough to extract the relational structure from natural language questions with high accuracy. Keyword-based query interfaces, on the other hand, are easy to use but have limited expressiveness and still suffer from the ambiguity inherent in the natural language terms used as keywords.

A second problem is that people have many different ways to express the same meaning, which can result in vocabulary and structure mismatches between the user's query and the machine's representation. This is often referred to as the semantic heterogeneity problem. Today we still heavily rely on ad hoc and labor-intensive approaches to deal with the semantic heterogeneity problem.

Third, the Web has seen increasing amounts of open-domain semantic data with heterogeneous or unknown schemas. Processing such data presents challenges to traditional NLI systems, which typically require well-defined schemas.

In this paper, we present our system to address these problems. We introduce a new schema-free query interface that we call the SFQ interface, in which the user explicitly specifies the relational structure of the query as a graphical "skeleton" and annotates it with freely chosen words, phrases and entity names. By using SFQ interface, we work around the unreliable step of extracting complete relations from natural language queries.

One motivation for our work is an enhancement to a system we are developing with RedShred, LLC that will help people identify and analyze business documents that include RFPs, RFQ, calls for proposals, BAAs, solicitations and similar business documents. Our prototype uses document analysis, information retrieval, NLP information extraction and question answering techniques and is largely domain independent. It understands general RFP-related concepts (e.g., proposal deadlines, duration, deliverables, security requirements, points of contacts, etc.) and can extract and organize information to help someone quickly evaluate opportunities. However, it does not have built-in knowledge of any particular domain, such as software development or material science, and is thus unable to address potentially critical characteristics involving them. For RFPs about software development, for example, we may need to know if the work requires a particular programming language (e.g., Java), is targeted for a given system or architecture (e.g., iOS), or has special requirements (e.g., 3DES encryption).

Given the breadth and variety of domains of interest, manually developing and maintaining custom ontologies, language models and systems for each is not viable. We are currently working on a system for automatic discovery of slots and fillers from RFP documents similar to infoboxes in Wikipedia and that are linked to DBpedia ontology. We plan to build on the results of this work to be able to provide schema free query support over extracted slots and fillers from RFP documents.

Our framework makes three main contributions. It uses robust methods that combine statistical association and semantic similarity to map user terms to the most appropriate classes and properties used in the underlying ontology. Second, it uses a novel type inference approach based on concept linking for predicting classes for

subjects and objects in the query. Third, it implements a general property mapping algorithm based on concept linking and semantic text similarity.

The remainder of the paper proceeds as follows. Section two describes our word similarity model. Sections three and four give an overview of our concept level association model trained on DBpedia and our query mapping approach. Sections five and six describe our enhancements to support the challenge queries and generating final SPARQL queries and in section seven we present the conclusions.

2 Semantic Similarity Component

We need to compute semantic similarity between concepts in the form of noun phrases, such as City and Soccer Club, and between relations in the form of short phrases, such as crosses and birth date. A common approach is using distributional similarity [HA68], which is a statistical approach that uses a term's collective context information drawn from a large text corpus to represent the meaning of the term. Distributional similarity is usually applied to words but it can be generalized to phrases [LI01]. However, the large number of potential input phrases precludes precomputing and storing distributional similarity data and computing it dynamically as needed would take too long. Thus, we assume that the semantic of a phrase is compositional on its component words and we apply an algorithm to compute semantic similarity between two phrases using word similarity.

We pair words from two phrases in a way such that it maximizes the sum of word similarities of the resulting word-pairs, similar to [MI06]. The maximized sum of word similarities is further normalized by the number of word-pairs. Computing semantic similarity between noun phrases requires additional work. Before running algorithm on two noun phrases, we compute the semantic similarity of their head nouns. If it exceeds an experimentally determined threshold we run the algorithm and if not, the phrases have similarity of zero. Thus we know that dog house is not similar to house dog.

Our word similarity measure is based on distributional similarity and latent semantic analysis, which is further enhanced using human crafted information from WordNet. Our distributional similarity approach, based on [RA03], yields a correctness of 92% on TOEFL synonym test, which is the best performance to date. By using a simple context of bag of words, the similarity between words even with different parts of speech can also be computed.

Although distributional similarity has an advantage that it can compute similarity between words that are not strictly synonyms, the human judgments of synonymy found in WordNet are more reliable. Therefore, we give higher similarity to word pairs which are in the same WordNet synset or one of which is a near hypernym of the other by adding 0.5 and 0.2 to their distributional similarities, respectively. We also boost similarity between a word and its derivationally related forms by increasing their distributional similarity by 0.3. We do so because a word can often represent the same relation as its derivationally related forms in our context. As examples, "writer" work as the almost same relation to "write" and so does "produce" to "product" be-

cause “writer” means the subject that writes and “product” means the thing being produced.

In our case, the lexical categories of words are not important and only their semantics matters. However, the value of distributional similarity of words is significantly lowered if they are not in the same lexical category. To counteract this drawback, we put words into the same lexical category using their derivational forms and compute distributional similarity between their aligned forms. Then we compare this value with their original similarity and use the larger one as their similarity.

The DBpedia ontology is a shallow ontology and many subclasses of Person class are not included. Consequently, it is possible that some person subtypes appearing in the user query have no similarity to any existing person class in the DBpedia ontology. To address this problem, we enforce a lower bound similarity, 0.25, between person and any person subtype so that these subtypes can at least be mapped to the DBpedia Person class.

We use WordNet to find whether a concept in the semantic graph is a person subtype or not. An ideal semantic similarity measure in our scenario should give high similarity to the terms that can work as synonymous substitution and low similarity to those not. The order of terms with high similarity score is not critical because statistical association can discriminate them and find the most reasonable one. Our implementation has been developed using this strategy. Semantic similarity is an active research field in natural language processing community and has been improved steadily over the years [LI98, HA13]. This component can be enhanced further to benefit from recent progress in this field.

3 Concept level Association Knowledge Model (CAK Model)

We use fully automatic approaches to obtain necessary domain knowledge for interpreting SFQs. Instead of a manually maintained lexicon, we employ a computational semantic similarity measure for the purpose of locating candidate ontology terms for user input terms. Semantic similarity measures enable our system to have a broader linguistic coverage than that offered by synonym expansion by recognizing non-synonymous terms that have very similar meaning. For example, the properties author of and college are good candidates for the user terms “wrote” and “graduated from”, respectively. Semantic similarity measures can be learned from a domain-dependent large corpus.

We know birds can fly but trees cannot and that a database table is not kitchen table. Such knowledge is essential for human language understanding. We refer to this as Concept level Association Knowledge (CAK). Domain and range definitions for properties in ontologies, argument constraint definitions of predicates in logic systems and schemata in databases all belong to this knowledge. However, manually defining this knowledge for broad or open domains is a tedious task at best. We therefore, learn Concept-level Association Knowledge statistically from instance data (the “ABOX” of RDF triples) and compute degree of associations between concepts based on co-occurrences. We count co-occurrences between schema terms indirectly from

co-occurrences between entities because entities are associated with types. We then apply a statistical measure, Pointwise Mutual Information (PMI) [CI07, DR10], to compute degree of associations between classes and properties and between two classes. The detailed approach is available in [HA12].

We used the learned CAK and semantic similarity measures for mapping a user query to a corresponding SPARQL query which we discuss in the next section.

4 Query Interpretation

In this section, we present the main steps in mapping terms in a SFQ to DBpedia ontology terms. The approach focuses on vocabulary or schema mapping, which is done without involving entities.

For each SFQ concept or relation, we generate a list of the k most semantically similar candidate ontology classes or properties. In the example in Figure 1, candidate lists are generated for the five user terms in the SFQ, which asks Which author wrote the book Tom Sawyer and where was he born?. Candidate terms are ranked by their similarity scores, which are displayed to the right of the terms.

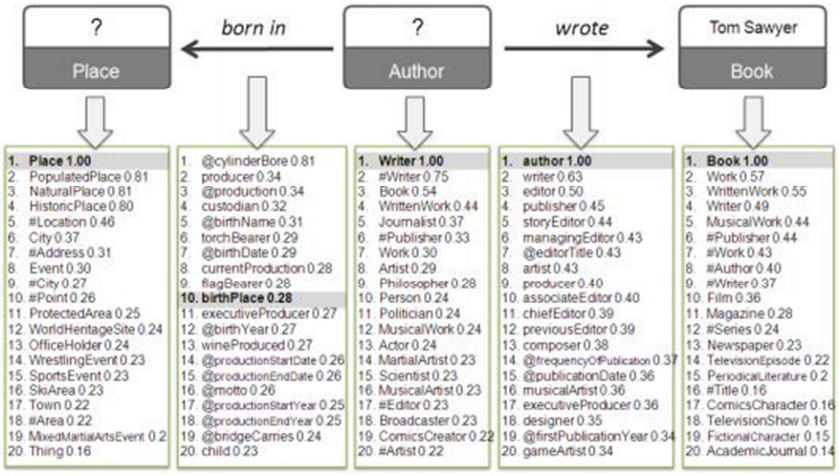

Fig. 1. A ranked list of candidate ontology terms

Each combination of ontology terms, with one term coming from each candidate list, is a potential query interpretation, but some are reasonable and others not. Disambiguation here means choosing the most reasonable interpretations from a set of candidates. An intuitive measure of reasonableness for an interpretation is the degree to which its ontology terms associate in the way that their corresponding user terms connect in the SFQ.

For example, since "Place" is connected by "born in" in Figure 1, their corresponding ontology terms can be expected to have good association. Therefore, the combination of Place and birthPlace makes much more sense than that of Place and @cylinderBore because CAK tells us that a strong association holds between Place and birthPlace but not @cylinderBore.

As you can see, we use the degree of association from CAK to measure reasonableness. As another example, CAK data shows that both the combinations of Writer + writer and Writer + author are reasonable interpretations of the SFQ connection "Author → wrote". However, since only author not writer has a strong association with the class Book, the combination of Writer, author and Book produces a much better interpretation than that of Writer, writer and Book for the joint SFQ connection "Author → wrote → Book".

We select two types of connections in a SFQ for computing the overall association of an interpretation. They are the connections between concepts and their relations (e.g., "Author" and "wrote") and the connections between direct connected concepts (e.g., "Author" and "Book"). We exclude indirect connections (e.g., between "Book" and "born in" or between "Book" and "Place") because they do not necessarily entail good associations.

If candidate ontology terms contained all the substitutable terms, we could rely solely on their associations for disambiguation. However, in practice many other related terms are also included and therefore the similarity of candidate ontology terms to the user terms is an important feature to identify correct interpretations. We experimentally found that by simply weighting their associations by their similarities we obtained a better disambiguation algorithm.

We use a linear combination of three pairwise associations to rank interpretations. The three are (i) the directed association from subject class to property (ii) the directed association from property to object class and (iii) the undirected association between subject class and object class, all weighted by semantic similarities between ontology terms and their corresponding user terms.

Our approach has a unique feature that it resolves mappings only using information in concept space, i.e., at the schema level. This makes it much more scalable than those that directly search into both instance and concept space for possible matches since concept space is much smaller than instance space.

5 Type Inference and Property Mapping

The SFQ system requires users to provide types or classes for subjects and objects in the query triples, however, this information is not available in many challenge queries. To bridge the gap we added a module for type inference for the challenge queries. The type inference is based on linking subject and object in the triple to Wikipedia concepts and retrieving the associated DBpedia ontology classes.

We use entity linking approach based on Wikitology [SY11] to link any named entities to concepts in Wikitology. We further enhanced Wikitology's entity linking system with gazetteers of named entities. For linking other topical concepts and key-

words we used Wikipedia Miner service [MI08]. Wikipedia Miner also links named entities, however when we tested with few examples we found Wikitology's named entity linking relatively more accurate and therefore we used Wikitology for named entity linking and Wikipedia Miner for linking other types of concepts. For Wikipedia Miner we used a probability threshold of 0.4. We tested with a lower threshold to improve recall but observed decrease in accuracy. For example, for the question "Which river does the Brooklyn Bridge cross?", the service predicted a link for "cross" to "http://en.wikipedia.org/wiki/Cross" which was not relevant. A threshold of 0.4 worked much better.

After linking the subject and object to concepts in Wikipedia we retrieve the associated DBpedia ontology classes. For named entities we detect the main type of named entity i.e. Person, Place or Organization based on associated DBpedia classes or mapped Schema.org classes. For example, for "Prince William, Duke of Cambridge" the associated type in DBpedia ontology is "BritishRoyalty" which is a subclass of "Royalty" which in turn is a subclass of "Person". We restricted to detecting main named entity types instead of fine-grained entity types as many entities in Wikipedia do not have a fine grained entity type associated with them. For other topical concepts we selected the most generalized class below the "Thing" class. For cases where the property values are literals, we fetch the matching property from DBpedia ontology and fetch the xsd type for the range of the property and map all numeric types such as integer, float etc. to "Number" type which is accepted by the SFQ system.

Table 1. Type Inference for Challenge Queries

Triples in Query	**Triples input to SFQ system (after type inference)**
?uri type Person . ?uri dbo:birthPlace res:Vienna . ?uri dbo:deathPlace res:Berlin .	?uri/**Person**, bornIn, Vienna/**Place** ?uri/**Person**, diedIn, Berlin/**Place**
?uri locatedOn Earth . ?uri type Mountain . ?uri height ?height .	?uri/**Mountain**, locatedOn, Earth/**CelestialBody** ?uri/**Mountain**, height, ?height/**Number**
Jane_Fonda marriedTo ?uri .	Jane_Fonda/**Person spouse** ?uri .

In additional to type inference we also try to map the user input property to DBpedia property based on linked concept. After linking the subject or the object to Wikipedia, we retrieve all associated DBpedia properties for that concept and compute similarity with the property input by the user based on the semantic text similarity module. For higher accuracy we only consider matching the property if the similarity score is at least 0.7. Table 1 shows examples of type inference and property mapping. The first example shows type inference for Vienna and Berlin to "Place". The

second example shows numeric type inference of height to “Number”. The third example shows property mapping from “marriedTo” to “spouse” using concept linking to *Jane_Fonda* and then retrieving the most similar property to the given property using semantic similarity.

6 SPARQL Query Generation and Selection

After user terms are disambiguated and mapped to appropriate ontology terms, translating a SFQ to SPARQL is straightforward. Figure 2 shows a sample SPARQL query produced by the system. Classes are used to type the instances, such as ?x a dbo:Writer, and properties used to connect instances as in ?0 dbo:author ?x. The bif:contains property is a built-in text search function which find literals containing specified text. The named entities in the SFQ can often be disambiguated by the constraints in the SPARQL query. In this example, Tom Sawyer has two constraints: it is in the label of some book and is written by some writer. For the challenge queries there were cases of aggregates, filtering and ordering. For such queries we explicitly appended the respective clauses to the SPARQL produced by the system before querying DBpedia.

```
PREFIX dbo:<http://dbpedia.org/ontology/>
SELECT DISTINCT ?x, ?y WHERE {
    ?0 a dbo:Book .
    ?0 rdfs:label ?label0 .
    ?label0 bif:contains '"Tom Sawyer"' .
    ?x a dbo:Writer .
    ?y a dbo:Place .
    {?0 dbo:author ?x} .
    {?x dbo:birthPlace ?y} .
}
```

Fig. 2. SPARQL Query generated by the system

Our Schema Free Querying system generates a ranked list of SPARQL queries. Some of the queries may not return results as the corresponding DBpedia instance may not have a property with the same name. For example, “mayor” is a valid property in DBpedia but for the case of Berlin, the property used is “leader”. In such cases the top ranked query may not return any results. Therefore, we iterate over ranked queries until we find a query that returns results from DBpedia.

7 Conclusions

The schema-free structured query approach allows people to query the DBpedia dataset without mastering SPARQL or acquiring detailed knowledge of the classes,

properties and individuals in the underlying ontologies and the URIs that denote them. Our system uses statistical data about lexical semantics and RDF datasets to generate plausible SPARQL queries that are semantically close to schema-free queries. We described our framework for handling schema agnostic or schema free queries and discussed enhancements to handle SAQ-2015 challenge queries. The key contributions of our approach are the robust methods that combine statistical association and semantic similarity to map user terms to the most appropriate classes and properties used in the underlying ontology and type inference for user input concepts based on concept linking.

8 References

[CI07] Philipp Cimiano, Peter Haase, and Jörg Heizmann. Porting Natural Language Interfaces between Domains: an Experimental User Study with the ORAKEL System. Proc. 12th Int. Conf. on Intelligent User Interfaces, pp. 180–189. ACM, 2007.

[DR10] Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. Entity Disambiguation for Knowledge Base Population, Proc. 23rd Int. Conf. on Computational Linguistics, August 2010.

[HA12] Lushan Han, Tim Finin, and Anupam Joshi, Schema-free Structured Querying of DBpedia Data, In Proc. 21st ACM International Conference on Information and Knowledge Management, pp. 2090-2093. ACM, 2012.

[HA13] Lushan Han, Tim Finin, Paul McNamee, Anupam Joshi and Yelena Yesha, Improving Word Similarity by Augmenting PMI with Estimates of Word Polysemy, IEEE Transactions on Knowledge and Data Engineering, IEEE Computer Society, v25n6, pp. 1307-1322, 2013.

[HA14] Lushan Han, Schema Free Querying of Semantic Data, Ph.D. Dissertation, University of Maryland, Baltimore County, August 2014.

[HA68] Zellig Sabbettai Harris. Mathematical Structures of Language. Wiley, New York, 1968.

[KA14] Abhay Kashyap, Lushan Han, Roberto Yus, Jennifer Sleeman, Taneeya Satyapanich, Sunil Gandhi and Tim Finin, Meerkat Mafia: Multilingual and Cross-Level Semantic Textual Similarity systems, Proc. 8th Int. Workshop on Semantic Evaluation, August 2014

[LI01] Dekang Lin and Patrick Pantel, Discovery of inference rules for question answering, Natural Language Engineering, 7(4):343–360, 2001.

[LI98] Dekang Lin. Automatic retrieval and clustering of similar words, Proc. 17th Int. Conf. on Computational Linguistics, pp. 768–774, Montreal, CN, 1998.

[MI06] Mihalcea, Rada, Courtney Corley, and Carlo Strapparava. Corpus-based and Knowledge-based Measures of Text Semantic Similarity. In Proc. 21st National Conf. on Artificial Intelligence, pages 775–780, 2006.

[MI08] David Milne and Ian H. Witten. "Learning to Link with Wikipedia." In Proceedings of the 17th ACM conference on Information and knowledge management, pp. 509-518. ACM, 2008.

[RA03] Reinhard Rapp, Word sense discovery based on sense descriptor dissimilarity, Proc. 9th Machine Translation Summit, pp. 315–322, 2003.

[SY11] Zareen Syed and Tim Finin, Creating and Exploiting a Hybrid Knowledge Base for Linked Data, in Agents and Artificial Intelligence, Revised Selected Papers Series: Communications in Computer and Information Science, v129, Springer, April 2011.