Schema-Free Structured Querying of DBpedia Data

Lushan Han
University of Maryland, Baltimore County
Baltimore, Maryland 21250
lushan1@cs.umbc.edu

Tim Finin
University of Maryland, Baltimore County
Baltimore, Maryland 21250
finin@cs.umbc.edu

Anupam Joshi
University of Maryland, Baltimore County
Baltimore, Maryland 21250
joshi@cs.umbc.edu

ABSTRACT

We need better ways to query large linked data collections such as DBpedia. Using the SPARQL query language requires not only mastering its syntax but also understanding the RDF data model, large ontology vocabularies and URIs for denoting entities. Natural language interface systems address the problem, but are still subjects of research. We describe a compromise in which non-experts specify a graphical query "skeleton" and annotate it with freely chosen words, phrases and entity names. The combination reduces ambiguity and allows the generation of an interpretation that can be translated into SPARQL. Key research contributions are the robust methods that combine statistical association and semantic similarity to map user terms to the most appropriate classes and properties in the underlying ontology.

Categories and Subject Descriptors

H.3 [**Information Systems**]: Information Storage and Retrieval; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

General Terms

Algorithms, Languages

Keywords

schema-free query, question answering, ontology mapping

1. INTRODUCTION

DBpedia [2], an RDF representation of information extracted from Wikipedia, is the key Linked Open Data (LOD) integrating component. It provides an open domain ontology in which each ontology class, property and instance is referenced by a unique URI that is trivially mapped to a Wikipedia article, faciliating linking data across domains and systems such as Freebase. and Facebook's Open Graph.

However, it is still difficult for typical Web users and even experts to query DBpedia with the standard RDF query language SPARQL. Since Wikipedia infoboxes are designed by different communities and edited by individuals, infobox names and attributes are largely heterogeneous. This heterogeneity is also a problem for the DBpedia ontology where terms with similar meaning are used in different contexts. For example, the property *locatedInArea* is used for mountains and *location* for companies. A second challenge comes from the large number of DBpedia vocabulary terms: currently 320 classes, 1,650 properties and several million instances. It is infeasible for users to be familiar with so many artificial terms. Finally, using SPARQL also requires mastering its complex syntax and semantics.

Allowing users to express queires in their own natural languages could help. Providing Natural Language Interfaces (NLIs) to structured data has been studied for more than four decades. Work in the 1970s and 1980s focused on developing NLIs to Databases (NLIDB) [1] and more recently on access to RDF [7]. The advantage of NLIs is that they do not require users to learn artificial syntaxes and vocabularies. However, there are two major obstacles for NLI systems to be widely adopted. First, current NLP techniques are still brittle when dealing with the ambiguity and complexity of Natural Language (NL) in general [1, 3]. Second, it requires extensive domain knowledge for interpreting NL questions. Domain knowledge typically consists of a *lexicon*, which maps user vocabulary to ontology vocabulary or logical expressions in NLIDBs, a *language model* with statistical information for parsing and sense disambiguation, and a *world model* specifying the relationships between the vocabulary terms such as subclass and the constraints on the types of properties arguments. Developing these can be very expensive, both for broad or open domains like DBpedia or a narrow domain like tax law.

We introduce a Schema-Free Query (SFQ) interface in which users specify a graphical "skeleton" for a query and annotate it with freely chosen words, phrases and entity names. An example is shown in Figure 1. The SFQ interface is a compromise between SPARQL and NLI. By asking users to specify semantic relations between entities in a query and the types of the entities, we avoid the difficult problem of relation extraction [3] from NL sentences. While the full expressive power of human language is not supported, people can use familiar vocabulary terms in composing a query.

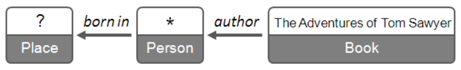

Figure 1: A Schema-Free Query for "Where was the author of the Adventures of Tom Sawyer born?".

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. In this paper, we introduce an approach to automatically learn this knowledge from semantically marked-up instance data.

With the learned CAK and semantic similarity measures, we present a straightforward but novel approach that disambiguates a SFQ and maps it to a corresponding SPARQL query to produce an answer. 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.

2. AUTOMATIC CAK LEARNING

We learn Concept-level Association Knowledge statistically from instance data (the "ABOX" of RDF triples) and thus avoid expensive human labor in building the knowledge manually. However, instead of producing "tight" assertions such as those used in RDF property domain and range definitions, we generate the *degree of associations*. Classical logics that make either true or false assertions are less suited in an open-domain scenario, especially those created from heterogeneous data sources. For example, what is the range of the property *author*? Both *Writer* and *Artist* are not appropriate because the object of *author* could be something other than *Writer* or *Artist*, e.g., *Scientist*. Specifying *Person* for the range is too general to be useful. Thus we use no a fixed range for the *author* property but a weighted set of classes capturing the likelihood of each being the type of an object.

Computing statistical association requires information on the number of occurrences of single terms and the number of co-occurrences of multiple terms in the universe. For DBpedia, the universe is represented by the dataset *Ontology Infobox Properties*, which contains RDF triples describing all relations between instances, and the dataset *Ontology Infobox Types*, which provides all type definitions for the instances. Figure 2 shows how we count term occurrences and co-occurrences by observing one relation in the universe. On the figure's left, is an RDF triple describing a relation and the type definitions for its subject and object and on the right are the resulting occurrences and co-occurrences of terms. We consider direction in counting co-occurrences between classes and properties. The directed co-

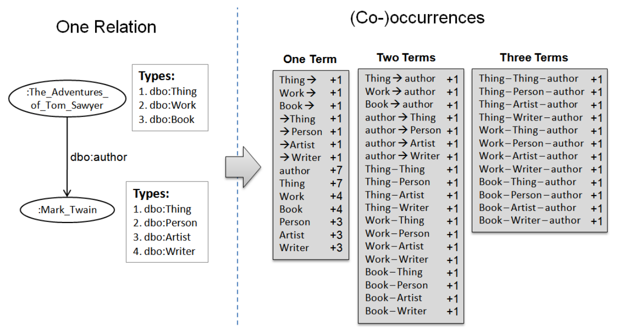

Figure 2: This example shows how we count term (co-)occurrences in an RDF knowledge base.

1) Writer→: @pseudonym 6.0, notableWork 6.0, influencedBy 5.7, skos:subject 5.7, influenced 5.5, movement 5.1, ethnicity 4.3, @birthName 4.3, @deathDate 4.2, relative 4.1, occupation 4.0, @birthDate 3.8, nationality 3.4, education 3.4, child 3.3, award 3.2, deathPlace 3.2, @activeYearsStartYear 3.2, partner 3.2, @activeYearsEndYear 3.1, genre 3.1, spouse 3.0, birthPlace 3.0, citizenship 2.9, foaf:homepage 2.8

2) →Writer: author 6.8, influencedBy 6.4, influenced 6.1, basedOn 5.3, illustrator 5.1, writer 5.1, creator 5.1, coverArtist 4.4, executiveProducer 4.4, relative 4.2, translator 4.1, lyrics 4.0, previousEditor 3.9, editor 3.6, spouse 3.5, child 3.4, nobelLaureates 3.3, designer 3.2, partner 3.2, associateEditor 3.2, director 3.0, narrator 3.0, chiefEditor 2.9, storyEditor 2.8, person 2.7

3) Book→: @isbn 5.8, @numberOfPages 5.8, @oclc 5.6, mediaType 5.6, @lcc 5.6, literaryGenre 5.6, @dcc 5.5, author 5.4, coverArtist 5.2, @publicationDate 5.1, nonFictionSubject 5.1, illustrator 5.1, translator 4.9, publisher 4.9, series 4.5, language 4.0, subsequentWork 3.3, previousWork 3.2, country 1.7, designer -1.9, @meaning -1.9, @formerCallsign -2.1, @review -2.4, @callsignMeaning -2.5, programmeFormat -2.6

4) →Book: notableWork 6.8, firstAppearance 6.4, basedOn 6.1, lastAppearance 5.9, previousWork 5.8, subsequentWork 5.8, series 4.8, knownFor 3.8, notableIdea 3.1, portrayer 2.6, currentProduction 2.3, related 1.9, author 1.7, nonFictionSubject 1.7, writer 1.4, translator 1.1, influencedBy 1.1, significantProject 1.1, award 0.9, coverArtist 0.8, relative 0.5, movement 0.5, associatedMusicalArtist 0.5, associatedBand 0.4, illustrator 0.3

5) author: →Writer 6.8, Musical→ 6.1, Play→ 5.4, Book→ 5.4, Website→ 5.4, WrittenWork→ 5.1, →Journalist 5.0, →Philosopher 4.9, →Website 4.8, →Artist 4.5, →Comedian 4.1, →Person 3.9, →ComicsCreator 3.8, →Scientist 3.6, TelevisionShow→ 3.4, Work→ 3.3, →Senator 3.2, →FictionalCharacter 2.8, →PeriodicalLiterature 2.7, →Governor 2.4, →Wrestler 2.3, →MemberOfParliament 2.3, →OfficeHolder 2.3, →Cleric 2.2, →MilitaryPerson 2.2

Figure 3: The top-25 most associated properties/classes from DBpedia's CAK or five examples.

occurrences are indicated by the arrow character → between two terms, for example *Book→author*. The occurrences of directed classes (e.g. *Book→*) are counted separately from the occurrences of undirected classes (e.g. *Book*).

After both occurrence and co-occurrence counts are available, we employ a statistical measure, Pointwise Mutual Information (PMI) [4, 6], to compute two types of associations: (i) directed association between classes and properties and (ii) undirected association between two classes.

Figure 3 shows examples of top-25 lists of most associated properties/classes for five terms along with their PMI values. Examples 1 to 4 present, in order, outgoing and incoming properties for two classes *Writer* and *Book*. Note that datatype properties are indicated by starting an initial '@' to distinguish them from object properties. Example 5 shows the classes that could be in domain or range of the property *author*. Terms ending and starting with → are potential domain and range classes, respectively.

In the first four examples, top properties are the most informative, such as *@pseudonym* and *notableWork* for *Writer* and *@isbn* and *@numberOfPages* for *Book*. More lowly ranked properties tend to be less related to the classes. Example 2 shows that both *author* and *writer* can be incoming properties of *Writer* though *author* is more related. On the other hand, the third example shows that only *author*, not *writer*, can describe *Book*. In the DBpedia ontology, *author* and *writer* are used for different contexts and *author* is used for books. The reason the class *Writer* has both *author* and *writer* as incoming properties is that writers can write something other than books, such as films and songs. Example 5 illustrates the DBpedia ontology's heterogeneity via the property *author*, which is loaded with multiple senses (e.g., book author, Web site creator). Noisy data in DBpedia can result in some abnormal associations, as shown in example 4, where *author* can be an incoming property of *Book*. Fortunately, the degree of these associations is typically low.

3. MAPPING APPROACH

In this section, we give 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.

3.1 Candidate Generation

For each SFQ concept or relation, we generate a list of the k most semantically similar candidate ontology classes or properties. (See [5] for our semantic similarity computation). A minimum similarity threshold, currently experimentally set at 0.1, is used to guarantee that all the terms have at least some similarity. For a relation that has very general meaning, such as "in", "has" and "from", we generate the $\frac{k}{2}$ ontology properties most semantically similar to each of its connected concepts because the semantics of a default relation is often conveyed in one of its connected concepts. We also generate $\frac{k}{4}$ ontology properties that are most semantically similar to the words *locate* and *own* on the behalf of "in" and "has", respectively. Finally we assemble these into a list of $\frac{3}{2}k$ ontology properties. The selection of a value for k is a compromise between the translation performance and the allowed computation time and depends on the degree of heterogeneity in the underlying ontologies and the fitness of the semantic similarity measure.

In the example in Figure 4, 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.

3.2 Disambiguation

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 4, 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,

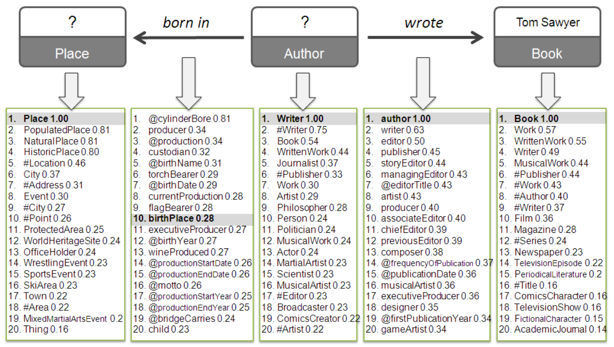

Figure 4: Lists of candidate ontology terms.

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.

To formalize our approach, suppose the query graph G_q has m edges and n nodes. Each concept or relation x_i in G_q has a corresponding set of candidate ontology terms Y_i. Our interpretation space H is the Cartesian product over the sets Y_1, ..., Y_{m+n}.

$$H = Y_1 \times ... \times Y_{m+n} = \{(y_1, ..., y_{m+n}) : y_i \in Y_i\}$$

Each interpretation $h \in H$ also describes a function $h(x)$ that maps x_i to y_i for $i \in \{1, ..., m+n\}$.

We define a fitness function $\Phi(h, G)$ that produces the fitness score of an interpretation h on a query graph or subgraph G. We seek the interpretation $h^* \in H$ that maximizes the fitness on the query graph G_q, which is computed as the summation of the fitness on each link L_i in G_q, i from 1 to m. More specifically,

$$h^* = \underset{h \in H}{argmax}\, \Phi(h, G_q) \tag{1}$$

$$\doteq \underset{h \in H}{argmax} \sum_{i=1}^{m} \Phi(h, L_i) \tag{2}$$

where link L_i is a tuple with three elements: subject concept s_i, relation r_i and object concept o_i. Formula 2 achieves *joint disambiguation* because the joint concepts of different links should be mapped to the same ontology class.

$\Phi(h, L_i)$ is the linear combination of three pairwise associations: the directed association from subject class $h(s_i)$ to property $h(r_i)$, the directed association from property $h(r_i)$ to object class $h(o_i)$, and the undirected association between subject class $h(s_i)$ and object class $h(o_i)$, all weighted by semantic similarities between ontology terms and their corresponding user terms. We need resolve the direction of $h(r_i)$ before we compute $\Phi(h, L_i)$ because $h(r_i)$ and r_i may have opposite directions. For approach details, please refer to [5].

If each candidate list contains k semantically similar terms, the computation complexity of a straightforward disambiguation algorithm is $O(k^{n+m})$ because the total number of interpretations is k^{n+m}. We can significantly reduce this complexity by exploiting locality. The optimal mapping choice of a property can be determined locally when the two classes it links are fixed. Thus we need only iterate on all combinations of classes, which have a total number k^n. Moreover, we can iterate in a way such that the next combination differs from current combination only on one class with other classes remaining unchanged. This enables us to re-compute only for the links in which the changed class participates and reuse previous computations on other links. The average number of links in which a class participates is $\frac{2m}{n}$. On the other hand, finding the property that maximizes the fitness of a link requires going through all k choices in the candidate list, resulting in $O(k)$ running time. Put them together, the total computation complexity can be reduced to $O(k^n \frac{m}{n} k)$. Further optimization can be achieved by decomposing the graph into subgraphs.

4. SPARQL GENERATION

After users terms are disambiguated and mapped to appropriate ontology terms, translating a SFQ to SPARQL is straightforward. Figure 5 shows the SPARQL query produced from the SFQ in Figure 4. 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 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.

```
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} .}
```

Figure 5: SPARQL query

5. EVALUATION

To evaluate ontology-based Question Answering systems, the 2011 Workshop on Question Answering over Linked Data [8] provided 50 training and 50 test questions over DBpedia 3.6 along with their ground truth answers. Of the 50 test questions, we selected 33 that could be answered using only the DBpedia ontology, i.e., without the additional assertions in the YAGO ontology.

Three computer science graduate students who were unfamiliar with DBpedia and its ontology independently translated the 33 test questions into SFQs. We first familiarized the subjects with the SFQ concept and its rules and then trained them with ten questions from the training dataset. Finally, we asked each subject to draw SFQs for the 33 test questions. None of the subjects had difficulty in constructing the SFQs and all finished within half an hour.

Three versions of 33 SFQs were given to our system which automatically translated them into SPARQL queries. The queries were then run on public SPARQL endpoints to produce answers. The answer of each query was evaluated for standard precision and recall. The average precision and recall of 33 queries on three versions were 0.754 and 0.832, respectively. The average time to translate a SFQ was only two seconds. Please refer to [5] for evaluation details.

6. CONCLUSION

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. An evaluation using an independently developed set of natural language queries showed that non-expert users were able to write effective SFQ queries for them with good accuracy.

7. ACKNOWLEDGMENTS

This research was partially supported by MURI award FA9550-08-1-0265 from the Air Force Office of Scientific Research.

8. REFERENCES

[1] I. Androutsopoulos, G. Ritchie, and P. Thanisch. Natural language interfaces to databases – an introduction. *Natural Language Engineering*, 1(01):29–81, 1995.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A Nucleus for a Web of Open Data. In *Proc. 6th Int. Semantic Web Conf.*, 2007.

[3] R. Bunescu and R. Mooney. A shortest path dependency kernel for relation extraction. In *Proc. of the HLT/EMLNP*, 2005.

[4] K. Church and P. Hanks. Word association norms, mutual information and lexicography. In *Proc. 27th Annual Conf. of the ACL*, pages 76–83, 1989.

[5] L. Han, T. Finin, and A. Joshi. Gorelations: Towards an intuitive query system for RDF data. Technical report, U. Maryland, Baltimore County, Feb. 2012.

[6] L. Han, T. Finin, P. McNamee, A. Joshi, and Y. Yesha. Improving word similarity by augmenting PMI with estimates of word polysemy. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, to appear.

[7] V. Lopez, V. Uren, M. Sabou, and E. Motta. Cross Ontology Query Answering on the Semantic Web: An Initial Evaluation. In *Proc. 5th Int. Conf. on Knowledge Capture*. ACM, 2009.

[8] 1st workshop on question answering over linked data. http://www.sc.cit-ec.uni-bielefeld.de/qald-1, 2011.