Preprint: Lushan Han, Tim Finin and Anupam Joshi, GoRelations: An Intuitive Query System for DBpedia, Proceedings of the Joint International Semantic Technology Conference, Springer LNCS, December 2011.

GoRelations: An Intuitive Query System for DBpedia*

Lushan Han, Tim Finin, and Anupam Joshi

University of Maryland, Baltimore County, USA
{lushan1,finin,joshi}@umbc.edu

Abstract. Although a formal query language, SPARQL, is available for accessing DBpedia, it remains challenging for users to query the knowledge unless they are familiar with the syntax of SPARQL and the underlying ontology. We have developed both an intuitive *semantic graph* notation or interface allowing one to pose a query by annotating a graph with natural language terms denoting entities and relations and a system that automatically translates the query into SPARQL to produce an answer. Our key contributions are the robust techniques, combining statistical association and semantic similarity, that map user terms to the most appropriate classes and properties used in the DBpedia Ontology.

Key words: Intuitive Query, Ontology Mapping, Statistical Association

1 Introduction

The growth of Linked Open Data (LOD) has made large amounts of Semantic Web data available. DBpedia [1] is an important example, since it is a key LOD integrating component serving as a microcosm for larger, evolving LOD collections. Most of DBpedia data is extracted from Wikipedia infoboxes, which are designed by different communities and edited by individuals, making infobox names and attributes largely heterogeneous. DBpedia addressed this problem by manually mapping infoboxes describing the same type of thing to the same DBpedia ontology class and synonymous attributes to the same ontology property, resulting in 272 classes and 1,300 properties as of DBpedia 3.6. However, heterogeneity remains a problem, especially for properties due to their large number and the difficulty of dealing with context-dependent mappings.

Although SPARQL is available for querying DBpedia, it remains difficult for typical Web users to query its knowledge base (KB). They must simultaneously master SPARQL, explore the large number of ontology terms, and deal with term heterogeneity. To simplify access, systems like True Knowledge and PowerAqua [2] provide natural language interfaces (NLIs) receiving users' queries and automatically finding answers in their underlying KBs. While they are good at answering simple questions (*Who were Richard Nixon's children?* and *Who*

* This research was supported in part by a gift from Microsoft, NSF award IIS-0326460 and the Human Language Technology Center of Excellence.

did Julie Nixon marry?) they often fail at slightly more complex ones (*Who did President Nixon's children marry?*) due to difficulties in understanding complex natural language (NL) questions.

GoRelations (*Graph of Relations*) is an open domain, intuitive query system that is easy to learn and use. It has two components: a *semantic graph interface* (SGI) allowing users to ask queries with complex relations and an effective and efficient automatic translator mapping the semantic graph into a corresponding SPARQL query to produce an answer. While our approach is tailored to DBpedia, the idea is generic and can be adapted to other LOD collections.

2 Semantic Graph Interface

Our interface uses an intuitive concept we call a *semantic graph* (SG) as a representation allowing a user to express a question or description. A semantic graph consists of nodes denoting entities and links representing binary relations between them. Each entity is described by two unrestricted terms: its name or value and its concept in the query context. Figure 1 shows an example of a semantic graph that comprises three entities: a place, person and book, which are linked by two relations, *born in* and *author*. Users flag entities they want to see in the results with a '?' and those they do not with a '*'.

Terms for concepts can be nouns (*book*) or simple noun phrases (*soccer club*) and relations can be references as verbs (*wrote*), prepositions (*in*), nouns (*author*) or simple phrases (*born in*). Users are free to name concepts and relations in their own ways as in composing a NL question with a current constraint that concept names should be at most two words and relation names three. One reason for the constraint is that most class and property names in the underlying DBpedia Ontology are no longer than two words and three words, respectively. A more fundamental reason is to encourage users to decompose their queries into simple entity and relation terms rather than use complex linguistic descriptions.

The value of entities can be something other than a name, for example, a number or date. If the value of an entity is a number, "Number" should be used as the entity's concept. Numerical attributes such as population, area, height, and revenue can be thought of as either relations or concepts, but since *Number* is already used as a concept, we require them to be relations. We enforce this rule because in DBpedia's ontology numerical attributes only have data types, which we uniformly treat as *Number* instances.

We circumvent the difficult task of understanding sentential semantics by asking users to directly supply the compositional relations between the lexical terms while users are not required to know a formal language and ontology.

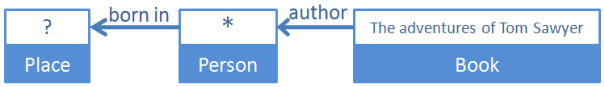

Fig. 1. A SG for "Where was the author of the Adventures of Tom Sawyer born?".

3 Translation

We start by laying out a novel, three-step approach that maps terms in the semantic graph to ontology terms. The approach focuses on vocabulary or schema mapping, which is done without involving entities. We then discuss how to generate a SPARQL query from the mappings. Finally, we describe the ontology statistics and semantic similarity components used in the mapping approach.

3.1 Mapping Approach

Step one: finding semantically similar ontology terms. For each concept or relation in the semantic graph, we generate a list of the k candidate ontology classes or properties that are most semantically similar. (See Section 3.4 for 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. If the relation is a very general term such as *in*, *has* and *from*, which we call "default relation", we generate $\frac{k}{2}$ most semantically similar ontology properties to each of its connected concepts. We do so because using one concept's name as relation name is the typical way to represent *has-relation* or *in-relation* in Wikipedia and because the semantics of a default relation is often conveyed in one of its connected concepts. The value k (currently 20) depends on the degree of heterogeneity in the underlying ontologies, how well semantic similarity measure is implemented, and the allowed computation time.

In Figure 2, candidate lists are generated for the five user terms in the semantic graph query. Classes starting with # are virtual classes[1], which we assign only half similarity to make them subordinate to native classes. Datatype properties are indicated by a starting @ character to distinguish them from object properties. Candidate terms are ranked by their similarity scores, which are displayed to the right of the terms. The example shows that our semantic similarity measure is well implemented but still has a large space to improve.

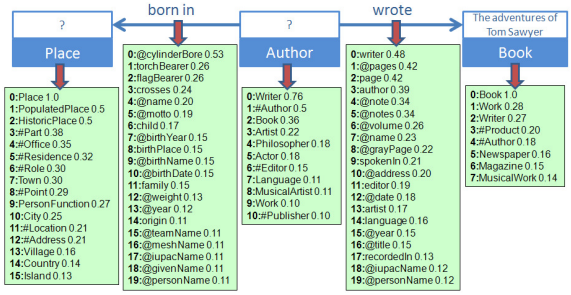

Fig. 2. Lists of candidate ontology terms.

Step two: disambiguation. Each combination of ontology terms, with one term coming from one candidate list, is a potential interpretation of the user query, but some are reasonable and others not. Disambiguation in this context means, among all the interpretations, chooses the most reasonable ones. An intuitive measure of reasonableness is the degree to which the ontology terms, in one interpretation, associate in the way that their corresponding user terms connect in the semantic graph. DBpedia is a knowledge representation of the

[1] They are inferred from the object properties (e.g. "publisher"). We create them to facilitate the mapping.

world's facts made by humans and a semantic graph is a description of some facts about the world in the user's mental model. Since both of them are mirrors of the world, they share an important feature – associations. Consider the example in Figure 2. In the query graph the relation *wrote* connects the two entities whose concepts under the query context are *Author* and *Book*. This implies that the relation *wrote* should have good associations with the concepts *Author* and *Book*. What should be reflected in the DBpedia Ontology is that the property corresponding to the relation *wrote* should also have good statistical associations with the classes corresponding to the concepts *Author* and *Book*.

Using association to resolve ambiguity is a common practice, as often seen in word sense disambiguation. However, many of previous researches only made use of coarse-grained association. That is, their contexts used for disambiguation are only a bag of words without considering the compositional structure of the knowledge in the sentences. With the coming of DBpedia, a large machine-readable KB, we are now able to compute fine-grained associations. We use Pointwise Mutual Information (PMI) [3] to compute statistical associations between classes and between classes and properties (See Section 3.3 for details).

If candidate ontology terms ideally contained all the near-synonyms, we could rely solely on their fine-grained 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 obtain a better disambiguation algorithm.

Below, we present a simple but novel disambiguation algorithm that exploits fine-grained associations. Suppose the query graph G has m links and n nodes. We need find a combination of m ontology properties p_1 to p_m, and n ontology classes, c_1 to c_n, from the space H of all interpretations that maximize the goodness or reasonableness of the mapping on the query graph G. This is computed as the summation of goodness of the mapping on each link L_i, i from 1 to m. More specifically,

$$\underset{\mathrm{p_1..p_m\ c_1..c_n \in H}}{\operatorname{argmax}} \mathrm{goodness(G)} = \underset{\mathrm{p_1..p_m\ c_1..c_n \in H}}{\operatorname{argmax}} \sum_{\mathrm{i=1}}^{\mathrm{m}} \mathrm{goodness(L_i)} \tag{1}$$

Note that the global optimal mapping on the whole graph is not necessarily composed of all the local optimal mappings on the individual links. Since a node can be involved in multiple links, the mapping decision on the node is affected by all the links it participates in. For example, the "Author" node in Figure 2 is involved in both the left link *born in* and the right link *wrote*. The local optimal mapping decision for "Author" from one link may be given up if it produces low goodness score on the other link. The same principle can be recursively spread to all other nodes and links in the entire graph. Therefore, our approach maps the semantic graph *jointly*.

Each link L_i is a tuple with three elements: subject concept S_i, relation R_i and object concept O_i. Let their corresponding ontology terms of current interpretation be $c(S_i)$, $p(R_i)$ and $c(O_i)$. Before we compute the goodness of link L_i,

we need first resolve the direction of the property $p(R_i)$ because $p(R_i)$ is semantically similar to R_i but they may have opposite directions. For example, the relation *wrote* in Figure 2 is semantically similar to the property *author* which, however, connects from *Book* to *Author*. We invent the statistical association measure $\overrightarrow{\text{PMI}}$ (see Section 3.3) to help determine the direction of $p(R_i)$. $\overrightarrow{\text{PMI}}$ measures statistical association between a class and a property. Unlike the traditional PMI, $\overrightarrow{\text{PMI}}$ also considers direction. $\overrightarrow{\text{PMI}}$(Class c, Property p) measures the strength of association between c as subject and p as predicate whereas $\overrightarrow{\text{PMI}}$(Property p, Class c) measures the strength of association between p as predicate and c as object. Whether the direction of $p(R_i)$ should be inverse to the one of R_i is decided in Formula 2.

$$\begin{aligned}
&\text{If } [\overrightarrow{\text{PMI}}(\text{c}(\text{O}_\text{i}), \text{p}(\text{R}_\text{i})) + \overrightarrow{\text{PMI}}(\text{p}(\text{R}_\text{i}), \text{c}(\text{S}_\text{i}))] \\
&\qquad - [\overrightarrow{\text{PMI}}(\text{c}(\text{S}_\text{i}), \text{p}(\text{R}_\text{i})) + \overrightarrow{\text{PMI}}(\text{p}(\text{R}_\text{i}), \text{c}(\text{O}_\text{i}))] > \alpha \\
&\quad \text{Then } S_i{}' = O_i,\ O_i{}' = S_i \\
&\quad \text{Else } \ S_i{}' = S_i,\ O_i{}' = O_i
\end{aligned} \tag{2}$$

The association term $\overrightarrow{\text{PMI}}(\text{c}(\text{O}_\text{i}), \text{p}(\text{R}_\text{i})) + \overrightarrow{\text{PMI}}(\text{p}(\text{R}_\text{i}), \text{c}(\text{S}_\text{i}))$ measures the degree of reasonableness of the inverse direction and the term $\overrightarrow{\text{PMI}}(\text{c}(\text{S}_\text{i}), \text{p}(\text{R}_\text{i})) + \overrightarrow{\text{PMI}}(\text{p}(\text{R}_\text{i}), \text{c}(\text{O}_\text{i}))$ measures the degree of reasonableness of the original direction. If the inverse direction is much more reasonable than the original direction, we inverse the direction by switching the classes that $p(R_i)$ connects; otherwise we respect the original direction. Currently, the reverse threshold α is 2.0. The Formula 2 worked very well empirically. Further verifying the formula and the hypothesis behind it using statistical techniques is one of our future work.

Finally, the goodness on link L_i is the sum of three pairwise associations: the directed association from subject class $c(S_i{}')$ to property $p(R_i)$, the directed association from property $p(R_i)$ to object class $c(O_i{}')$, and the undirected association between subject class $c(S_i{}')$ and object class $c(O_i{}')$, all weighted by semantic similarities between ontology terms and their corresponding user terms.

$$\begin{aligned}
\text{goodness}(\text{L}_\text{i}) = \max(&\overrightarrow{\text{PMI}}(\text{c}(\text{S}_\text{i}{}'), \text{p}(\text{R}_\text{i})) \cdot \text{sim}(\text{S}_\text{i}{}', \text{c}(\text{S}_\text{i}{}')) \cdot \text{sim}(\text{R}_\text{i}, \text{p}(\text{R}_\text{i})) \\
&+\overrightarrow{\text{PMI}}(\text{p}(\text{R}_\text{i}), \text{c}(\text{O}_\text{i}{}')) \cdot \text{sim}(\text{O}_\text{i}{}', \text{c}(\text{O}_\text{i}{}')) \cdot \text{sim}(\text{R}_\text{i}, \text{p}(\text{R}_\text{i})), \beta) \\
&+\text{PMI}(\text{c}(\text{S}_\text{i}{}'), \text{c}(\text{O}_\text{i}{}')) \cdot \text{sim}(\text{S}_\text{i}{}', \text{c}(\text{S}_\text{i}{}')) \cdot \text{sim}(\text{O}_\text{i}{}', \text{c}(\text{O}_\text{i}{}'))
\end{aligned} \tag{3}$$

We use a parameter β (currently 0.05) to shield the effect of the first two pairwise terms on the occasions when the property $p(R_i)$ fits too poorly with its two classes to be a valid choice (their value can be negative infinite). For these cases, the goodness is determined only by the last pairwise term.

Of the best interpretation yielded by the disambiguation algorithm for the example in Figure 2, the concepts *Place*, *Author* and *Book* are mapped to the ontology classes *Place*, *Writer* and *Book* respectively. The relation *born in* and *wrote* are mapped to the ontology property *birthPlace* and *author* with direction unchanged and reversed respectively. Although the property *writer* has larger

semantic similarity to the user term *wrote* than the property *author*, it is not selected because it is mainly used for describing films or songs but rarely for books. Although the property *birthPlace* has relatively low similarity with *born in*, it is selected because all the candidate terms with higher similarity do not associate well with the classes similar to the concept *Place* and *Author*.

The computation complexity of a straightforward disambiguation algorithm is $O(k^{n+m})$ simply because the total number of interpretations is k^{n+m}. However, we can significantly reduce this complexity to $O(k^n \frac{m}{n} k)$ by exploiting locality because the optimal mapping choice of a property can be determined locally when the two classes it links are fixed.

Step three: refinement. The best interpretation typically gives us the most appropriate classes and properties that the user terms can map to. However, for properties there can be some issues requiring additional work. First, the concepts are mapped to correct classes but occasionally the relation connecting them cannot find any mapping. Second, although the disambiguated property is appropriate, sometimes it is not the major property used in the context. Because the concepts are already disambiguated, we can narrow down to the disambiguated context where we can have more information about all properties that actually connect the two known classes and their conditional probabilities. In the case of a missing property, we map the relation to its most semantically similar property in the context. In the case of a minor property, we add other properties in the context, which are less similar to the user relation than the disambiguated property but have much higher conditional probabilities.

3.2 SPARQL Generation

After users terms are disambiguated and mapped to appropriate ontology terms, the translation of a semantic graph query to SPARQL is straightforward. Figure 3 shows the SPARQL query produced for the semantic graph in Figure 2. Classes are used to type the instances, such as *?x a dbo:Writer*. Properties are used to connect instances just as relations do for entities as in *?0 dbo:author ?x*. If a user relation is mapped to multiple properties, the SPARQL UNION operator is used to combine them. *bif:contains* is a virtuoso built-in text search function which find literals containing specified text.

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

Fig. 3. SPARQL Query Generated

3.3 Ontology Statistics Component

GoRelations uses three kinds of ontology statistics: directed association between classes and properties, undirected association between classes themselves, and conditional probability of properties given two connected classes. Computing

these statistics requires information about the number of occurrences of a term and the number of co-occurrences of two or three terms in the universe consisting of all relations. In 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.

The example in Figure 4 explains how we count term occurrences and co-occurrences by observing one relation in the universe. On the left of the figure, we give an RDF triple describing a relation and the type definitions for the subject and object in the triple. On the right, we list the resulting occurrences and co-occurrences of terms. The directed co-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*).

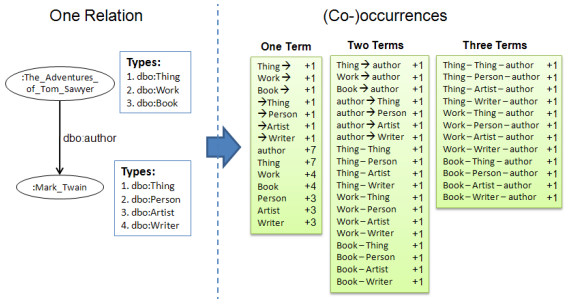

Fig. 4. An example for counting (co-)occurrences

PMI is used to measure the strength of statistical association between two terms. Equation 4 gives the PMI formula where f_{t_1} and f_{t_2} are the marginal occurrence counts of the two terms t_1 and t_2 and $f(t_1, t_2)$ is the co-occurrence count of t_1 and t_2 in the universe. N is a constant for the size of the universe. $\overrightarrow{\text{PMI}}$ is computed the same way as PMI except that its class term is directed.

$$\text{PMI}(t_1, t_2) \approx \log_e\left(\frac{f(t_1, t_2) \cdot N}{f_{t_1} \cdot f_{t_2}}\right) \tag{4}$$

3.4 Semantic Similarity Component

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. As in Mihalcea's approach [4], we pair up words from two phrases in a way such that it maximizes the sum of word similarities of the resulting word-pairs. We differ, however, in that we do not allow a constituent word to participate in multiple pairs. The maximized sum of word similarities is further normalized by the number of words in the longer phrase to get the output similarity for two phrases.

Our word similarity measure is based on distributional similarity and latent semantic analysis, which is further enhanced using knowledge from WordNet. The distributional similarity approach that we use is described in [5] while we give higher similarity to word pairs which are in the same WordNet synset or one of which is the immediate hypernym of the other.

4 Evaluation

To evaluate ontology-based QA systems, the 2011 Workshop on Question Answering over Linked Data (QALD) provided 50 training and 50 test questions on DBpedia dataset along with their true answers. We use the 50 QALD training questions to tune our system, setting the thresholds and coefficients. Of the 50 test questions, 33 questions can be answered using only the DBpedia Ontology, while the rest need knowledge from the YAGO ontology. We used the 33 questions to evaluate our system. We modified seven of them that required Boolean answers or operations not supported by our semantic graph notation for the time being, such as grouping and counts. Our changes included changing the answer type or removing the unsupported operations but preserving the relations and thus the question schemata. Our collection of 33 test questions with their true answers are available at `http://ebiq.org/r/326`.

Three human subjects unfamiliar with the DBpedia ontology independently translated the test questions into semantic graph queries. Three versions of 33 semantic graphs were given to our system which automatically translated them into SPARQL queries. The average time to translate a semantic graph to its corresponding SPARQL queries was 0.38 second. The queries were then run on public DBpedia SPARQL endpoints to produce answers. The precision, recall and f-measure of our system, averaging on three versions, are 0.687, 0.722 and 0.704, respectively.

5 Conclusion

GoRelations is an intuitive query system that allows people to query DBpedia without mastering SPARQL or acquiring detailed knowledge of the classes and properties used in the underlying ontologies. It's interface uses a simple *semantic graph* notation for queries that is automatically translated into a corresponding SPARQL query. We developed a novel three-step mapping approach that disambiguates user terms in a semantic graph query and maps them to DBpedia ontology terms. Our system was evaluated on 33 QALD test questions and the result shows the approach works decently well.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. In: Proc. 6th Int. Semantic Web Conf. (2007)
2. Lopez, V., Fernndez, M., Motta, E., Stieler, N.: Poweraqua: Supporting users in querying and exploring the semantic web content. Semantic Web Journal (2011)
3. Church, K., Hanks, P.: Word association norms, mutual information and lexicography. In: Proc. 27th Annual Conf. of the ACL. (1989) 76–83
4. Mihalcea, R., Corley, C., Strapparava, C.: Corpus-based and knowledge-based measures of text semantic similarity. In: Proc. 21st AAAI Conf. (2006) 775–780
5. Rapp, R.: Word sense discovery based on sense descriptor dissimilarity. In: Proc. 9th Machine Translation Summit. (2003) 315–322