Semantic Search in Linked Data: Opportunities and Challenges

Hamid Haidarian Shahri
Department of Computer Science, University of Maryland, College Park, MD, USA
hamid@cs.umd.edu

Introduction [1]

In this abstract, we compare semantic search (in the RDF model) with keyword search (in the relational model), and illustrate how these two search paradigms are different. This comparison addresses the following questions: (1) What can semantic search achieve that keyword search can not (in terms of behavior)? (2) Why is it difficult to simulate semantic search, using keyword search on the relational data model? We use the term *keyword search*, when the search is performed on data stored in the relational data model, as in traditional relational databases, and an example of keyword search in databases is [Hri02]. We use the term *semantic search*, when the search is performed on data stored in the RDF data model. Note that when the data is modeled in RDF, it inherently contains explicit typed relations or semantics, and hence the use of the term "semantic search." Let us begin with an example, to illustrate the differences between semantic search and keyword search.

Semantic Search

Consider that we want to know more about "Michael Jordan." This entity of type *Person*, could be the *Professor*, who teaches *Computer Science*, and is affiliated with *UC Berkeley*. It could also be the *Basketball Player*, who plays for the *Chicago Bulls*, and is in the *NBA league*. A close analogy from the unstructured web of documents is that a Google search for "Michael Jordan" returns hundreds of pages, most of which are irrelevant to the Berkeley Professor. Most of the results and the top ranked ones refer to the *Basketball Player*, which may not be our intended entity for the search. Although the use of additional terms like "Berkeley" will help us in finding our intended entity, we may not know which university he is affiliated with. We might actually be performing the search to find this piece of information. Figure 1 demonstrates the two different entities and the information related to these entities, in the RDF model. The nodes represent the entities. The edges represent the properties, which hold between the entities.

With semantic search, in the RDF model, users can iteratively refine their search; navigate through the initial results and filter out the results (entities), which do not have the properties that they are looking for. In fact, the explicit representation of properties in RDF (which does not exist in the relation model) facilitates this refinement of search results. In Figure 1, the user could search for "Michael Jordan" and the instances, which match the search string, will be shown to the user. Now, since the user knows that he is looking for a *Professor*, he could select the *teaches* property from all the available properties, which refines the search to the entities that have a *teaches* property. This way, the *Professor* entity, which he is looking for, is found. If the user does not know what Michael Jordan teaches, he can find out the answer to this question by seeing the value for the *teaches* property, which is *Computer Science*. On the other hand, if he knows this fact, he can add the *teaches* property and the *Computer Science* property value, to further refine the result of the search, if necessary.

Intuitively, humans specify their intended entities in this fashion. In other words, they define an entity by iteratively specifying extra properties about an entity, until the desired entity is uniquely identifiable, for example, "Michael Jordan," the one who teaches Computer Science, and is affiliated with UC Berkeley, etc. Subclasses and other properties help in the search refinement process. Moreover, once the desired entity is uniquely identified, we can browse its various unknown properties, depending on what property we are looking for. In essence, RDF data browsers enable users to navigate through "sets of entities of the same type" and gradually refine these sets.

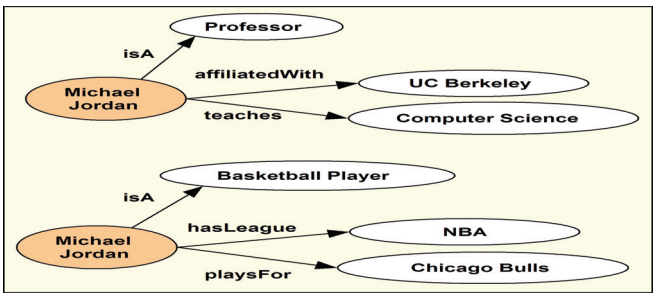

Fig. 1. Two different entities with the same name "Michael Jordan," represented in the RDF data model. Each entity has a different set of properties and property values.

Keyword Search

With keyword search, in the relational model, it is very difficult to perform the semantic search process, which was described in the Semantic Search section. This difficulty is in part due to the fact that the semantics are not encoded explicitly in the relational model (refer to the extended version). Consequently, it is difficult to incorporate metadata information (i.e. column names) into the keyword search process, as described below.

While recent research in the database literature has attempted to retrofit keyword search onto relational databases,

[1] The extended version of this abstract is available at: http://www.cs.umd.edu/~hamid/AAAI10PosterFull.pdf

there are various scalability issues in performing keyword search. For now, ignoring the computational cost (which will be discussed later), in theory, a keyword search like "Michael Jordan" and "Computer Science" is possible. However, this keyword search can only be performed, when we assume that the user knows the property values (i.e. Computer Science) that would sufficiently refine the search. Clearly, this is not always the case. In other words, the user can not browse the available properties (like *teaches*) and navigate through sets of entities, as in semantic search. Notice that in keyword search, unlike semantic search, we are not dealing with the *teaches* property, and instead need to use *Computer Science*, which is a property value for the *teaches* property.

In addition to this limitation in navigation, performing keyword search on top of relational databases is computationally expensive, especially when the keywords (i.e. property values) appear in various tables, or there is a long list of keywords, since this requires many joins. In general, the keyword search process requires various steps, including: finding keys in tables, finding joinable attributes, generating foreign key join candidates, and removing semantically incorrect candidates. Moreover, enumerating all possible candidate networks, which may contribute to the results, is computationally expansive [Hri02]. On the other hand, in semantic search, user's knowledge of the domain can be utilized effectively to navigate through sets of entities, and refine the search results. In fact, this "user-driven" navigation in semantic search, replaces the enumeration of candidate networks in keyword search (which is computationally expensive). In other words, while it is *unacceptable* to ask the user to provide a series of joins and SQL operations that are necessary for finding the correct results in the relational model, it is quite *acceptable* to ask the user to provide additional properties and property values to refine the search, as described in the Semantic Search section.

Another issue with performing semantic search on the relational model is related to physical implementation. The physical implementation of most relational databases follows their logical description, i.e. each table (relation) is stored in its own file, or collection of files, on disk. Such an implementation is effective for queries that filter, or aggregate, large portions of a single table. It provides reasonable performance for queries that join many tuples from one table, to another table [Mar08]. However, this implementation is much less effective for semantic search, which requires join queries that follow paths from a small number of tuples of one table, to another table. Note that semantic search queries try to accumulate facts about a small set of entities (e.g. all the cities in a country). Answering such queries requires one, or more, random I/Os for each table that is used in the path. Therefore, semantic search queries perform poorly on the traditional physical implementation in the relational model.

Aside from the explicit encoding of semantics, another advantage of the RDF model is the global *referencing* of entities on the entire web, which does not exist in the relational model. For example, the "Michael Jordan" entity, who is a basketball player, is specified by a link (or URI) and can be uniquely referenced by different organizations, across the web. Now assume that different kinds of information about the health information and the financial information of "Michael Jordan" are stored in different organizations. With semantic search in RDF, a user can access all this information from various organizations. Note that we are not designing or planning for any specific queries in advance, when the health and financial organizations are being constructed. In other words, the aggregation of information is achieved in an ad hoc manner. The global referencing of entities in RDF is vital to facilitating interoperability and aggregation/reuse of knowledge across organizational boundaries. Consider that in the relational model, facilitating interoperability across distributed, and heterogeneous databases is quite difficult, which is partly due to the lack of such a referencing mechanism.

Research Challenges

Storage and retrieval of data in the relational database model has become highly optimized, over the last three decades. Similar performance levels are necessary for RDF to enable large-scale semantic search. Considering that the representation of data in the RDF and relational models are different (refer to the extended version), there are fundamental database research issues that need to be studied in this area. The semantic search example described in the Semantic Search section, clarifies some of these issues, which include: native storage mechanisms for RDF or efficient storage of RDF in the relational model, indexing and retrieval of RDF data, optimization of queries specified in SPARQL, and ranking of entities in search results.

From a human-computer interaction standpoint, there are several issues that need to be studied to enable semantic search, including: effective presentation of sets of entities, user interactions to support the selection of some of the entities from the result of a query, presentation of relationships - at both class and instance level, and intuitive interfaces for specifying SPARQL queries (perhaps similar to query-by-example models). Each of these issues becomes clearer, when considered in the "context" of the precise examples, provided in this abstract, which describe the desired semantic search behavior.

The comparison in this abstract clarified the advantages of using RDF, instead of the traditional relational data model. We demonstrated that it is difficult to retrofit a robust search and find answers to questions about an entity (as available in RDF), on the relational data model. Note that the explicit encoding of semantics (via typed relations), and the global referencing of entities in RDF (via links or URIs) are the two critical enabling *features* that make RDF suitable for robust search and information integration across different enterprises. The comparison also revealed some of the crucial research challenges that need to be addressed for scalable semantic search.

References

[Hri02] Hristidis, V., Papakonstantinou, Y., "DISCOVER: Keyword Search in Relational Databases," *Proc. of VLDB'02,* Hong Kong, China, August 20-23, 2002.

[Mar08] Marcus, A., "BlendDB: Blending Table Layouts to Support Efficient Browsing of Relational Databases," MSc Thesis, MIT, 2008.