iRDQL - Imprecise Queries Using Similarity Joins for Retrieval in Ontologies

Anonymized

Abstract

Traditional semantic web query languages support a logic-based access to the semantic web offering a retrieval of data based on facts. On the traditional web and in databases, however, exact querying oftentimes provide incomplete answers as queries are over-specified or the mix of multiple ontologies/modelling differences requires "interpretational flexibility." This paper introduces iRDQL, a semantic web query language with support for similarity joins. It is an extension to RDQL that enables the user to query for similar resources in an ontology. In the context of an OWL-S matchmaking test collection we show that iRDQL is indeed useful for extending the reach of the query improving recall without sacrificing too much precision.

1 Introduction

Imagine the following situation: you want to buy a used car, which has certain properties such as a minimum age, color, etc. When executing the query on a semantically annotated database of cars, however, you are buried in hundreds of results (or you may not get any answer as you over-specified the query). This situation is very typical. People querying the semantic web, databases, or also the web in general oftentimes find themselves either buried in results to their queries or with no results whatsoever. A common approach to handle these problems is to rank the results of a query, in the case of too many answers, or to return similar results, when no precise matches to the query exist [Baeza-Yates and Ribeiro-Neto, 1999]. To achieve the same goal for the semantic web *we extended RDQL* [Seaborne, 2002], a query language for RDF [Lassila and Swick, 1999], *with similarity joins* [Cohen, 2000] *to retrieve not only the precise results of a query but also similar ones.* Thus, our approach, called iRDQL for *imprecise* RDQL, exploits the semantic annotation on the semantic web in conjunction with a similarity measure to improve the ranking of the results of queries for such resources. Hence, similar results may be found in the case where no precise results to a query exist. Additionally, if too many results are found, the intention of iRDQL is to use similarity measures to improve the ranking of the results. The *i* (*imprecise*) indicates that two compared resources do not have to be precisely equal but can be considered as imprecisely equal (or *similar*) according to some similarity measure.

2 iRDQL: RDQL with Similarity Joins

RDQL (Resource Description Query Language) [Seaborne, 2002] is a query language to formulate queries over RDF in Jena [Reynolds *et al.*, 2005]. For example, like SQL it allows to formulate a query that will retrieve all services (resources) that have profiles which exactly match a service called *Beach Surfing Profile* (the result being the *Beach Surfing Service*). But what if a user would like to find all services that have a profile similar to the *Beach Surfing Profile* to get a larger variety of services? To that goal we extended the RDQL language with two additional language constructs `IMPRECISE` and `SIMMEASURE`. The `IMPRECISE` clause defines the variables of the query whose bindings (found resources) should be matched imprecisely when executing the query. That is, they are added to the result set of the query together with their corresponding similarity value as computed by the similarity measure. The measure to compare two resources is specified by the `SIMMEASURE` clause. Here any similarity measure implemented in our generic Java library of similarity measures in ontologies can be used [citation suppressed]. The query corresponding our users desiderata would look as follows:

```
SELECT      ?S1,?P1,?P2
WHERE       ?S1 presents ?P1
            ?P2 serviceName "beach surfing"
IMPRECISE   ?P1,?P2
SIMMEASURE Levenshtein
```

The query looks for a service `?S1` with the profile `?P1` and retrieves all profiles `?P2` that have the service name *"beach surfing"*. It then computes the similarity between `?P1` and `?P2` returning the following result table:

S1	P1	P2	Sim
Beach Surfing Service	Beach Surfing Profile	Beach Surfing Profile	1.0
Beach Broker Service	Beach Broker Profile	Beach Surfing Profile	0.5
...	...	...	...

3 Experimental Evaluation

For our evaluation we chose the `OWL-S-TC-v1` [Klusch, 2005] OWL-S service retrieval test collection, which specifies a set of 406 OWL-S [Martin *et al.*, 2004] services and 9 queries with their "correct" answers to evaluate service

matchmaking algorithms. For each query, we generated two Levenshtein-based [Levenshtein, 1966] iRDQL statements: one applying the similarity join to the *service profile* (called *iP*) and a second one applying it to the *service profile as well as the service's process model* (called *iPM*). We executed each resulting iRDQL query against all the services belonging to the same domain and ranked the results according to the similarity (or average similarity, in the second case).

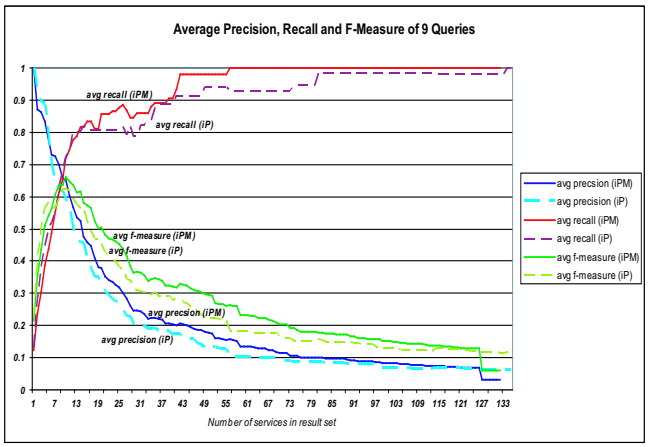

Figure 1: Precision, recall, and f-measure for all queries averaged for each query style (*iP* vs. *iPM*)

Figure 1 shows the average precision, recall, and the f-measure for each of the query types (*iP* vs. *iPM*). Here the x-axis expresses the n top-ranked services of the iRDQL-query's result set. The precision of the iRDQL query is 1.0 for a result set of size one indicating that on average all of the services inside that result set are correct answers. The overall trend of the precision is decreasing since the result set is constantly growing until its size reaches the total number of services of the domain. The recall of the iRDQL query constantly increases as additional services are added to the result set of the query.

The behavior observed in figure 1 illustrates the usefulness of our approach. In each of the domains, an exact query (i.e., an RDQL query with no similarity extension) yields exactly one result: the only perfect match. While this result has 100% precision it has a rather poor recall. The use of the imprecision extension for RDQL allowed us to simply extend the reach of the query and find additional correct matches without (at least initially) overly decreasing recall. Actually, as a comparison of both query styles shows, an increased use of similarity operators leads to better retrieval performance: *iPM* (which has two similarity joins) significantly outperforms *iP* (only one similarity join) on all measures as shown by a t-test (precision: $1.4e^{-19}$, recall: $9.8e^{-21}$, f-measure: $5.7e^{-19}$). Obviously, this evaluation of iRDQL can only serve as an illustration. A thorough evaluation will have to consider: (1) evaluate the approach's robustness towards the use of different similarity measures [citation suppressed], (2) explore different combination approaches for multiple similarity measures, and (3) investigate the computational consequences of using similarity joins over precise joins.

4 Conclusions

In this paper we introduced our approach of extending RDQL with similarity joins to find not only precise matches to a query but also a set of similar matches. Our implementation was inspired by Cohen's approach [Cohen, 2000] of using similarity joins to solve the problem of combining information from different databases. He uses a standard *tf-idf* scheme [Baeza-Yates and Ribeiro-Neto, 1999] to compute the similarity between columns from different tables. The main difference to our approach is that we are not dealing with flat tables (i.e., data in first normal form) but with complex (ontologized) objects (i.e., data stored in NF^2—non first normal form [Schek and Scholl, 1986]). This calls for a deeper investigation of similarity measures that rely on the structure of an ontology. Thus, further research is necessary to find the best performing similarity measure for the combination of iRDQL and OWL-S semantic web service descriptions. In accordance to Cohen's work we believe that the approach presented in iRDQL may provide the basis for combining the strengths of logic-based precise querying and similarity-based retrieval.

References

[Baeza-Yates and Ribeiro-Neto, 1999] R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval*. ACM Press, 1999.

[Cohen, 2000] William W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Trans. Inf. Syst.*, 18(3):288–321, 2000.

[Klusch, 2005] Matthias Klusch. OWLS-TC-v1: OWL-S service retrieval test collection. http://projects.semwebcentral.org/projects/owls-tc/, 2005.

[Lassila and Swick, 1999] Ora Lassila and Ralph R. Swick. Resource description framework (RDF) model and syntax specification. http://www.w3.org/TR/REC-rdf-syntax/, 1999.

[Levenshtein, 1966] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, vol. 10:707–710, 1966.

[Martin *et al.*, 2004] David Martin, Mark Burstein, Jerry Hobbs, and Ora Lassila. OWL-S: Semantic markup for web services. http://www.w3.org/Submission/OWL-S/, November 2004.

[Reynolds *et al.*, 2005] Dave Reynolds, Andy Seaborn, and Chris Dollin et al. Jena–A semantic web framework for java. http://jena.sourceforge.net/index.html, 2005.

[Schek and Scholl, 1986] H J Schek and M H Scholl. The relational model with relation-valued attributes. *Inf. Syst.*, 11(2):137–147, 1986.

[Seaborne, 2002] Andy Seaborne. RDQL–A query language for RDF. http://jena.sourceforge.net/tutorial/RDQL/index.html, 2002.