Scalewelis: a Scalable Query-based Faceted Search System on Top of SPARQL Endpoints

Joris Guyonvarch and Sébastien Ferré

IRISA, Université de Rennes 1
Campus de Beaulieu, 35042 Rennes cedex, France
joris.guyonvarch@gmail.com, ferre@irisa.fr

Abstract. This paper overviews the participation of Scalewelis in the QALD-3 open challenge. Scalewelis is a Faceted Search system. Faceted Search systems refine the result set at each navigation step. In Scalewelis, refinements are syntactic operations that modify the user query. Scalewelis uses the Semantic Web standards (URI, RDF, SPARQL) and connects to SPARQL endpoints.

1 Introduction

We participated with Scalewelis in the question answering task of the QALD-3 open challenge for DBpedia. This task was about answering a hundred questions formulated in several languages. The answer could be given either as a SPARQL query or directly as a list of results. SPARQL is a very expressive language that is based on graph patterns and relational algebra but it requires an important background for those who want to use it. We discuss two alternatives to SPARQL that are Question Answering and Faceted Search.

Question Answering (QA) gives answers to natural questions or keywords rather than from formal queries. For example, Querix [6] asks for users to write full English questions and, then, there is a dialog between Querix and users that clarifies ambiguities. However, question answering does not prevent dead ends, i.e. empty results.

Faceted Search guides users with feedback. The result set is incrementally refined by the selection of facets. A facet selection always leads to non-empty results, so that there are no dead ends. For example, VisiNav [4] is a set-based Faceted Search system that applies operations to the result set. gFacet [5] is a graph-based Faceted Search system that applies operations to the result graph. Sewelis [2] is a query-based Faceted Search system that applies operations to the query.

Scalewelis[1] is inspired from Sewelis. Sewelis manages its own data structures and does not scale to large datasets such as DBpedia. On the contrary, Scalewelis connects to SPARQL endpoints and uses partial result sets in order to scale to large datasets.

[1] Scalewelis has been implemented as a Web application and is available at: http://lisfs2008.irisa.fr/scalewelis/

```
SELECT DISTINCT ?class WHERE {
   [] a ?class . }
```

Fig. 1. Computation of classes at the initial step.

Query 1: Computation of the partial results, limited to 1,000 entities

```
SELECT DISTINCT ?result WHERE {
   <Pattern>
} LIMIT 1000
```

Query 2: Computation of class facets from the partial results

```
SELECT DISTINCT ?class WHERE {
   VALUES (?result) {res1 ... resN}
   ?result a ?class }
```

Query 3: Computation of property facets from the partial results

```
SELECT DISTINCT ?prop WHERE {
   VALUES (?result) {res1 ... resN}
   ?result ?prop [] }
```

Query 4: Computation of inverse property facets from the partial results

```
SELECT DISTINCT ?invProp WHERE {
   VALUES (?result) {res1 ... resN}
   [] ?invProp ?result }
```

Fig. 2. Computation of partial results and facets.

2 Approach

Query-based Faceted Search allows for more expressive search than traditional set-based Faceted Search. In Sewelis and Scalewelis, facets are query elements and selecting a facet adds it to the search query. Only valid facets are provided, so that navigation is safe: there are no dead ends. Facets are computed from the results of the previous query. Scalewelis uses only a subset of those results. In experiments on datasets of up to 15 millions triples, automatically generated with the Berlin SPARQL Benchmark [1], the partial computation of facets required far less time than the complete computation. Moreover, it nonetheless gave all the facets in most cases [3].

At the initial step, Scalewelis retrieves all classes in the dataset (Figure 1). Properties can be computed in a similar manner but are not in the current implementation for efficiency reasons. In the next steps, partial results and facets are computed with four successive SPARQL queries (Figure 2). Query 1 retrieves the first 1,000 results matching `<Pattern>`, which is the SPARQL graph pattern translated from the user query. Queries 1, 2, and 3 incorporate those results in

order to retrieve the facets for the current query, using the `VALUES` construct of SPARQL. Query 2 retrieves the types of the results, i.e. classes. Query 3 retrieves the properties linking *from* the results. Query 4 retrieves the property linking *to* the results, i.e. inverse properties. Finally, the selection of a facet by the user modifies the query so that it is more constrained but is ensured to return results.

3 Resources

Regarding the QALD-3 open challenge, Scalewelis was connected to the standard DBpedia SPARQL endpoint because response times of the provided endpoint were too long for effective use. The process is intrinsically interactive and we acted as the user. We answered 70 questions out of 99, requiring a few minutes per question (human and machine time). Two kinds of questions were not addressed. Firstly, questions involving quantities (such as *Which German cities have more than 250000 inhabitants?*) because filters and aggregations are not yet implemented in Scalewelis. Secondly, questions for which facets were not found by the user. In general, it was unclear whether those facets were not listed (due to partial results) or whether the user was not knowledgeable enough to identify the right facet (for example *Give me all B-sides of the Ramones.*).

One of the question in the challenge is *Which films starring Clint Eastwood did he direct himself?*. We explain how we answered this question with Scalewelis. At the initialization step, the user filtered the list of classes with the word *film*. The selection of the class *a dbo:Film* led to the query *What is a dbo:Film?*. At this step, there are property facets and the selection of *has dbo:starring* and *has dbo:director* led to the query *What is a dbo:Film and has dbo:starring a thing and has dbo:director a thing?*. Finally, the selection of the result *Clint Eastwood* at the two undefined element represented by *a thing* led to the final user query: *What is a dbo:Film and has dbo:starring dbr:Clint_Eastwood and has dbo:director dbr:Clint_Eastwood?*.

4 Results

Scalewelis ranked third out of six candidates with 32 correct answers plus 1 partial answer. Scalewelis was theoretically able to correctly answer the 70 questions the user chose to answer, but he failed to answer 37 of them because incorrect facets were chosen.

1. Firstly, English is not the mother tongue of the user so that incorrect facets were used because of misunderstandings.
2. Secondly, the user had sometimes the choice between two facets differing only by their namespace, for example DBpedia ontology and infobox properties.
3. Thirdly, the standard DBpedia SPARQL endpoint to which Scalewelis was connected sometimes gave different results compared to the provided SPARQL endpoint.

5 Discussion

Scalewelis can be improved in several ways. First, there are missing functionalities from SPARQL such as comparisons or aggregations. Second, the partial computation of results and facets entails a bias in the search, because the first results returned by SPARQL may not form a representative sample of all results. Third, Scalewelis could take advantage of the DBpedia ontology to further improve scalability, and to display facets as class/property hierarchies.

References

1. Bizer, C., Schultz, A.: The Berlin SPARQL Benchmark. International Journal on Semantic Web and Information Systems (IJSWIS) 5(2), 1–24 (2009)
2. Ferré, S., Hermann, A.: Reconciling faceted search and query languages for the semantic web. IJMSO 7(1), 37–54 (2012)
3. Guyonvarch, J., Ferré, S., Ducassé, M.: Scalable query-based faceted search on top of SPARQL endpoints for guided and expressive semantic search (2013), research report, IRISA
4. Harth, A.: VisiNav: A system for visual search and navigation on web data. J. Web Sem. 8(4), 348–354 (2010)
5. Heim, P., Ertl, T., Ziegler, J.: Facet Graphs: Complex semantic querying made easy. In: et al, L.A. (ed.) ESWC (1). pp. 288–302. Lecture Notes in Computer Science, Springer (2010)
6. Kaufmann, E., Bernstein, A.: Evaluating the usability of natural language query languages and interfaces to semantic web knowledge bases. J. Web Sem. 8(4), 377–393 (2010)