A Faceted Search Index for OptiqueVQS

Vidar N. Klungre and Martin Giese

University of Oslo

Abstract. We present an end-user oriented visual query system that combines faceted search with graph queries. Unlike previous systems, the architecture is designed to scale gracefully to both very large datasets *and* complex queries. This is achieved by setting up an indexing structure for facet values that can easily be scaled out arbitrarily. In return, it compromises some precision in computing sets of available facet values, but it does so in a highly configurable manner.

Keywords: Faceted search, RDF, Index, Visual Query Interface

1 Introduction

Faceted search [6] is a popular search and exploration paradigm which allows users to apply search filters to multiple orthogonal dimensions (facets) of the data. Filters can be added, removed or modified in any order, and every time this is done, the system immediately updates the list of results, giving the user instant feedback. To support this functionality, the system needs fast access to the underlying data. This is often provided by search engines like Lucene[1] or Sphinx[2] which provide better performance for the queries required by faceted search than RDB-based implementations.

Faceted search has also been extended to semantics-based data, e.g. in SemFacet [1] and Rhizomer [2]. In these systems, datatype properties are treated as facets, but they also include some ability to query the graph structure via object properties. Combining faceted search with graph queries results in more expressive queries, but implementing faceted search also becomes computationally harder. The challenging task is to update the set of available values for the facets after each user interaction. The straightforward way of computing this set involves evaluating a query of similar complexity to the whole query built so far, and that needs to be done for each facet. For queries with large graph patterns, and large datasets, this can become too time consuming for an interactive system. The usual approach of using a search engine style index will not help, since these engines do not support graph queries.

In this demo, we present a system that combines faceted search with graph queries, and that uses an indexing structure for facet values that can easily be scaled out arbitrarily. In return, it compromises some precision in computing sets of available facet values, in a highly configurable manner.

[1] `https://lucene.apache.org/`
[2] `http://sphinxsearch.com/`

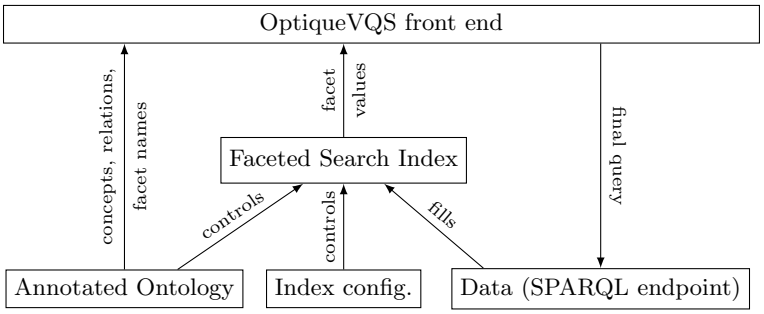


Fig. 1. Architecture of OptiqueVQS extended with the faceted search index.

We implemented this new functionality as part of OptiqueVQS [5], an ontology-based visual query system, intended for users with little IT expertise. OptiqueVQS combines graph querying with filtering on datatype properties. However, in previous versions of OptiqueVQS, the available values for each facet have been static, determined entirely by a suitably annotated ontology, and did not depend on the underlying data. Neither did they change in reaction to the filters on other facets. This new version adds a server side component that reads data from a SPARQL endpoint and stores information needed for efficient faceted search in a scalable index. This index, instead of the original SPARQL endpoint, is queried in reaction to user interactions to update the interface.

The essence of the index structure is to use one table (that could be stored e.g. in a Lucene index) per concept available in the query interface. In addition to the facet values applicable to that concept, the index can also store information about the existence of object property links to other resources, facet values of such neighbouring resources, links from those to further resources, etc. How much information about neighbouring resources is to be stored in the index can be configured. Independently of the configuration, the index will provide an approximation from above (a superset) of the possible remaining values for each facet.

2 Approximating Facet Value Computation

Fig. 1 shows the architecture of OptiqueVQS with the new faceted search index and its configuration in the centre. This component was missing from previous versions, where the only contact between OptiqueVQS and the actual data was when the finished query was executed. Now the essential information for presenting a reactive faceted search interface is pre-computed and stored in an index structure, which the front end communicates with.

Given a partially constructed query Q and a focus variable $?x$ in Q, the index has to answer the question which values v for a given datatype property p on $?x$ are still possible. In other words: all values v such that extending Q with a tuple $?x\ p\ v$ would not make the query unsatisfiable. Answering this exactly will in general be as expensive as evaluating the whole partial query.

We therefore approximate this computation, possibly computing a superset of the possible values, by considering only a part of Q, a neighbourhood of the focus variable $?x$. In the simplest case, object properties are ignored, and classical faceted search on the data properties for the variable is performed, using only data filters on $?x$. To do this effectively, an index table with one column per facet is created. Standard search engine technology can perform the required filtering and gathering of possible values in a scalable way.

However, usability studies with OptiqueVQS have shown that users react to facet values that they consider to be obviously excluded by some of the object properties in their partial query. To accommodate an object property o, we can add a boolean-valued facet representing the *existence* of a triple $?x\ o\ ?y$ to some resource $?y$ in the data. The index data structure remains a simple table of facet values. As an example, consider the following data:

```
:kona a :Drink; :kind "Coffee"; :suppliedBy :americanDrinks .
:sencha a :Drink; :kind "Tea"; :suppliedBy :asianDrinks .
:orangeJuice a :Drink; :kind "Juice" .
:americanDrinks a :Supplier; :basedIn "America" .
:asianDrinks a :Supplier; :basedIn "Asia" .
```

The user builds a query for drinks supplied by companies based in America, see screenshot (a) in Fig. 2. In screenshot (b), the index is configured not to take the suppliedBy relation into account. Therefore, all three kinds of drinks are available as options for the Kind facet. In (c), the existence of a supplier is added to the index, and the choice Juice is disabled, since there is no product of that kind that has a supplier.

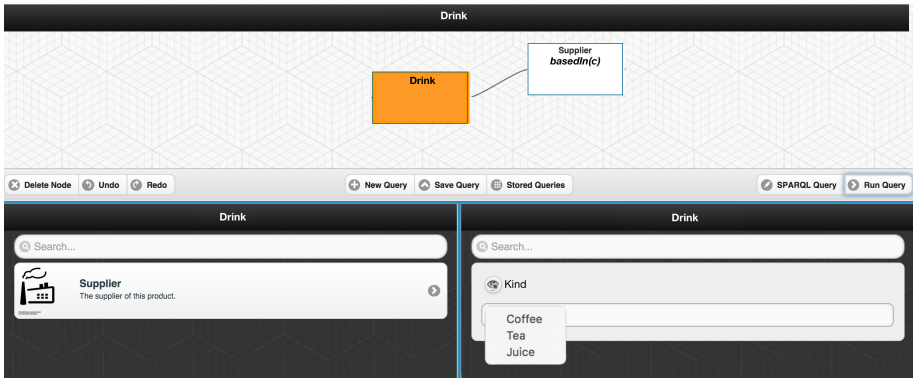
This can be further refined by adding columns for datatype properties on the resources connected by object properties. These can then be taking into account when filtering for combinations of values permitted by Q. In the example, adding the based in facet of the supplier to the index table for drinks means that only Coffee will be available, as screenshot (d) shows.

Which combinations of object and datatype properties should be added as columns to the search engine table, depending on the type of $?x$, is controlled by the index configuration. The details of this configuration, how it controls index generation, and how the index is used to compute sets of possible facet values is explained in a technical report, together with a more elaborate example [3].

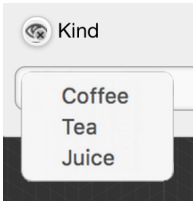
Independently of the configuration, the index has the shape of a single table that can be scaled out using standard methods to arbitrarily large datasets. If necessary, index storage and processing can be parallelized, which is considerably easier for single tables than for relational or graph databases.

We will demonstrate our tool on an instance of the NPD benchmark [4].

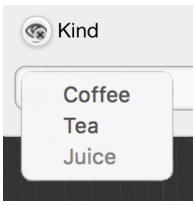
An interesting remaining question concerns the relationship between perceived usability and index configuration. The larger the neighbourhood considered by the index, the sharper the approximation of possible facet values, but the more space will also be required by the index. We surmise that users expectations of possible choices can be met by considering rather small configurations. To show this empirically in usability studies, and to get an idea of the trade-off between perceived usability and index size, are future work.



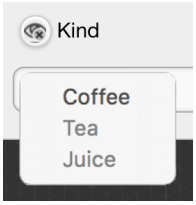
(a) example query



(b) without supplier



(c) existence of supplier



(d) location of supplier

Fig. 2. Screenshots of the system suggesting values.

Acknowledgements This work has been funded by the Norwegian Research Council through SIRIUS (NFR 237898).

References

1. Marcelo Arenas et al. Faceted search over RDF-based knowledge graphs. *J. Web Semantics*, 37:55–74, 2016.
2. Josep Maria Brunetti, Roberto García, and Sören Auer. From overview to facets and pivoting for interactive exploration of semantic web data. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 9(1):1–20, 2013.
3. Vidar N Klungre. A faceted search index for graph queries. Technical Report 469, Department of Informatics, University of Oslo, 2017. `http://heim.ifi.uio.no/martingi/pub/IndexReport.pdf`.
4. Davide Lanti, Martin Rezk, Mindaugas Slusnys, Guohui Xiao, and Diego Calvanese. The npd benchmark for obda systems. In *Proc. of the 10th Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2014)*, volume 1261 of *CEUR Workshop Proceedings*, pages 3–18, 2014.
5. Ahmet Soylu et al. Experiencing OptiqueVQS: a multi-paradigm and ontology-based visual query system for end users. *UAIS*, 15(1):129–152, 2016.
6. Daniel Tunkelang. Faceted search. *Synthesis lectures on information concepts, retrieval, and services*, 1(1):1–80, 2009.