SPARQL Faceter—Client-side Faceted Search Based on SPARQL

Mikko Koho, Erkki Heino, and Eero Hyvönen

Semantic Computing Research Group (SeCo),
Aalto University, Department of Computer Science, Finland
first.last@aalto.fi, http://seco.cs.aalto.fi/

Abstract. The faceted search paradigm is widely used in web applications, and there are various tools available for implementing it on the server side. In contrast, this paper presents an HTML based component tool on the client side that can be plugged on virtually any public SPARQL endpoint on the web, using *only* SPARQL API for data retrieval. To test and demonstrate the idea and the tool, application of the tool in a large in-use semantic portal is presented.

1 Introduction

Faceted search [2, 12], known also as view-based search [9] and dynamic hierarchies [10], is based on indexing data items along orthogonal category hierarchies, i.e., facets (e.g., places, times, document types etc.). In searching, the user selects in free order categories on facets, and the data items included in the selected categories are considered search results. After each selection, a count is computed for each category showing the number of results, if the user next makes that selection. In this way, search is guided by avoiding annoying "no hits" results. The idea of faceted search is especially useful on the Semantic Web where hierarhical ontologies used for data annotation provide a natural basis for facets, and reasoning can be used for mapping heterogeneous data to facets [4].

Faceted search can be implemented with popular server-side solutions, such as Solr[1], Sphinx[2], and ElasticSearch[3]. However there is a lack of light-weight client-side faceted search tools or components that search data directly from a SPARQL endpoint. Such a tool would be very useful, because it could be used on virtually any open SPARQL endpoint on the web without any need for server side programming and access rights. This paper presents a web component for implementing faceted search applications in a browser, based only on a standard SPARQL API.

[1] http://lucene.apache.org/solr/
[2] http://sphinxsearch.com/blog/2013/06/21/faceted-search-with-sphinx/
[3] https://www.elastic.co/

2 The Tool: SPARQL Faceter

Design Requirements The requirements for our tool, *SPARQL Faceter*, are based on our earlier experience on tediously implementing server side faceted search over and over again in different applications, as well as on the generic requirements of the search paradigm [12, 3]: there is a need for a lightweight browser-based faceted search engine tool that is easy to use and adapt for different RDF datasets published in SPARQL data services on the web. A particular demand of our own Linked Data Finland data service[4] [5] is to support external users of the data in application development, and here the notion of the Rich Internet Application, where the data service is completely detached using standard SPARQL API was deemed very useful. The tool should therefore fulfill the following requirements:

1. **Data read from a SPARQL endpoint.** It is a common practice to publish RDF datasets as SPARQL services, and build applications that use the SPARQL endpoint to query for information. The data can reside anywhere on the web and be managed by anybody on the server side. No extra effort is required for managing the data from the application developer side.
2. **Easily adaptable to different datasets.** Different datasets have different data models and the application should be easy to configure for these.
3. **Easy integration to web pages.** A developer should be able to integrate a faceted search on some web page and modify the appearance of the faceted search freely.
4. **Fluent user experience.** When a user makes selections of the facets, the application should show results in a reasonable time. The UI should be easy to use and should support the exploratory analysis of datasets.
5. **Support for hierarchical facets.** The tool should be able to handle hierarchical concepts in the facets, displaying the hierarchy in the facet selections. When a user selects a category upper in a hierarchy, the results should include all results for its' sub categories.
6. **Easy to maintain.** The tool is planned to be actively used, and thus should be easy to maintain and develop further.

In short, the goal was to create a tool that would be easy to use out-of-the-box, and require minimal configuration, yet provide enough configuration options to be useful in different contexts.

Design AngularJS[5] was chosen as the implementation framework for our tool. *SPARQL Faceter* is developed as open source with source code available on GitHub.

SPARQL Faceter consists of two distinct components. One is the main *Semantic Faceted Search* component[6], which contains the basic functionalities of

[4] http://ldf.fi
[5] https://angularjs.org/
[6] https://github.com/SemanticComputing/angular-semantic-faceted-search

the tool, and is dependent on the second component: an AngularJS service for querying SPARQL endpoints with paging and object mapping[7].

The *Semantic Faceted Search* component is comprised of different services, and most importantly, the facet selector directive. This directive is the main component of the tool, and provides the user interface for using the faceted search.

The facet selector directive is independent of any results the facet selections might imply: it's role is to keep track of what values are selected in what facets, and to display available selections to the user including the number of results each selection implies. The tool communicates the facet selections to whatever application is using it by calling a callback function whenever the selections change. Thus, *SPARQL Faceter* places no restrictions on what can be done with the user's selections. The tool does, however, provide services for integrating the facet selections to SPARQL queries, handling SPARQL results, and updating the URL based on current selections.

The *Semantic Faceted Search* component works by building a single SPARQL query for the facets it controls, and updating the query each time the user makes a selection. This has the advantage of keeping the states of the facets synchronized at all times. The downside is that with very many facets querying can become slow, and the user experience may suffer as the facets are unusable while their states are being queried. In order to make it feasible to have many facets available, facets can be enabled and disabled: only enabled facets are included in the query. Whether or not a facet is initially enabled is configurable. The tool shows a spinner on the components when querying for data, to convey to the user that the information in the elements is being updated.

For enabled facets, the tool shows all possible values for the facets' property, and the amount of instances that the selection results in. The amount of results is calculated based on all the current selections in all of the facets. Values that would lead to no results are omitted. In addition to the list of possible values, there is also a text input for searching the facet values.

The tool also supports hierarchical facets, in which selecting a category upper in the hierarchy also shows results for all the categories that are below it in the hierarchy. The hierarchy specification is not limited to using a predefined property or vocabulary, but is based on configuring a property path that captures the hierarchy, and giving the top categories explicitly. The facet element displays a two-level hierarchy of the categories. For categories below the current category level, the hierarchy is flattened and all the sub categories are listed below the upper category. The category level is initially on the topmost categories, and selecting a value changes category level to the level of the selected value.

The proposed way to use the facet selections, as returned by the tool, is to build another SPARQL query for retrieving the results according to the user's selections. By using the tool's services, the results can then be mapped into JavaScript objects with pagination and client-side caching of the received results.

[7] https://github.com/SemanticComputing/angular-paging-sparql-service

Installation and Configuration At its simplest, the only configuration options needed by *SPARQL Faceter* are the URL of a SPARQL endpoint, a callback function to be called with the facet selections when they change, the RDF class URI of the resources which are the target of the search, and the facet configurations. A basic facet can be configured with just the URI of the property and a name for the facet to be displayed to the user. The callback function defines what happens when the facet selections are updated, which typically is to update a results display according to user selections.

Other types of facets require some additional configuration options, but the amount of configuration needed for any facet type has been kept to a minimum. The other facet types include a keyword search facet, a time-span facet, and a hierarchical facet. More details about the configuration is given in the repository of the *Semantic Faceted Search* component.

As for the results, the user of the tool is expected to build a query for retrieving results based on the facet selections. This makes the tool extremely flexible to different data models, but requires that the user of the tool is familiar with SPARQL syntax.

3 Applications

We tested and evaluated *SPARQL Faceter* on the WarSampo portal [6]. The portal[8] consists of different application perspectives built on top of interlinked Finnish Second World War data published on a SPARQL endpoint. The SPARQL endpoint is provided by an Apache Fuseki Server[9]. The applications are tested to work with all major browsers (Chrome, Firefox and Internet Explorer).

Use case 1: Death Records Perspective The WarSampo death records perspective[10] uses the application to provide an interface for searching and exploring the death records of Finnish WW2 casualties [7]. The dataset consists of about 95,000 death records, and almost 2.4 million RDF triples.

Fig. 1 shows a screenshot of the faceted search of death records using *SPARQL Faceter*. The application interface contains 12 facets and a table-like view of the results. The facets include a keyword search facet for the persons' names, a time-span facet for the time of death and a hierarchical military rank facet, and 9 basic facets of the annotated properties of the death records. The source code of the application is available online[11].

A dataset this large provides a benchmark for the performance of the *SPARQL Faceter*, as SPARQL queries and also data handling on the client-side could take long enough to deteriorate the user experience. Quite much effort had to be put on optimizing the SPARQL queries for speed, while also making sure that no

[8] http://www.sotasampo.fi/
[9] https://jena.apache.org/documentation/fuseki2/
[10] http://www.sotasampo.fi/en/casualties/
[11] https://github.com/SemanticComputing/WarSampo-death-records

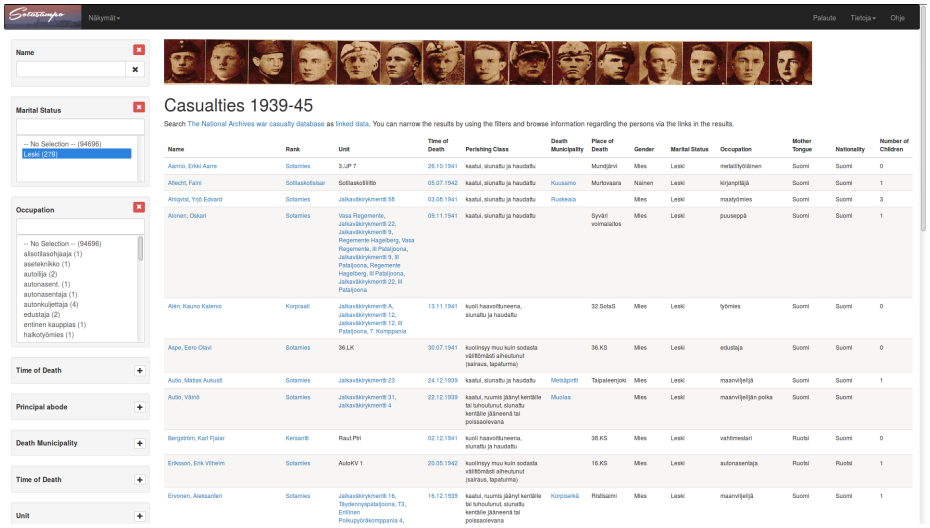

Fig. 1. The faceted search interface of death records [7] with a selected value in one facet.

unnecessary processing of the results happens on the client-side. Normally when searching and browsing the dataset, results are displayed in a few seconds after the user makes a selection. With some selections that result in a large result set, and if additionally many facets are enabled, the user may have to wait more than ten seconds to see the results and updated facets. Most of the time goes to waiting for results from the SPARQL endpoint.

Use case 2: Photographs Perspective The WarSampo photographs perspective[12] uses the application to create a faceted search interface for wartime photographs. The dataset consists of about 159,000 photographs and a total of about 1.6 million triples. The interface contains only 5 facets, which ensures fast response times. The facets include a time-span facet for the date the photograph was taken, a keyword search for descriptions, and basic facets for related people and places. Fig. 2 shows a screenshot of the photograph perspective.

The perspective uses "infinite scrolling" to display the photographs: more results are retrieved as the user scrolls down for more photographs. Clicking on a photograph shows the user more information about the photograph, and provides hyperlinks to other WarSampo perspectives via linked entities.

4 Discussion

In this paper we have presented the *SPARQL Faceter* for creating client-side faceted search applications. The configuration of the facets has been kept as

[12] http://www.sotasampo.fi/en/photographs/

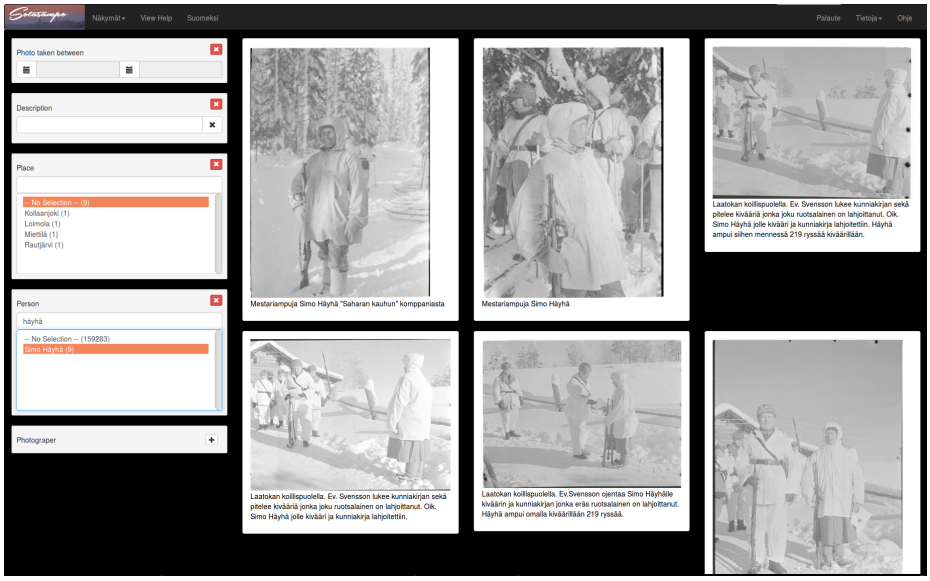

Fig. 2. The faceted search interface of wartime photographs with a selected value in one facet.

simple as possible. *SPARQL Faceter* handles querying the SPARQL endpoint when needed, mapping the SPARQL results to JavaScript objects, and creating and updating the facet elements. To make use of the facet selection values, the developer using the tool has to retrieve results based on selected values, which usually includes writing SPARQL queries.

Results Section 2 presented the design requirements for a Semantic Web tool that provides client-side faceted search of RDF datasets published as SPARQL services. Here is a detailed view of how the *SPARQL Faceter* manages to satisfy these requirements:

1. **Data read from a SPARQL endpoint.** All data is retrieved via SPARQL, from an endpoint defined by the user.
2. **Easily adaptable to different datasets.** Configuration of the tool is required for every dataset. The configuration required for setting up facets is kept to a minimum.
3. **Easy integration to web pages.** A faceted search can be easily added to a web page by using the developed AngularJS components. The facet selections have a default look that can be customized using CSS.
4. **Fluent user experience.** For datasets that are distinctly smaller than the ones used in the use cases, the tool is almost certainly fast enough for good user experience. For the faceted search applications in the use cases, some user selections can be slow, but common usage gives results based on

user selections in a few seconds. For larger datasets, the amount of facets may have to be restricted, depending on the processing capabilities of the SPARQL server.
5. **Support for hierarchical facets.** Two-level hierarchical facets are supported by the tool, and demonstrated in the first use case. The dataset can contain more than two levels, but the hierarchical facet flattens the hierarchy to two levels for display in the select element.
6. **Easy to maintain.** The separation of concerns design principle is used in development to simplify code structure and maintenance. Automated tests are used to help maintain good code quality.

Our application is able to satisfy the given requirements, and thus provides a usable tool for faceted search of RDF datasets.

Related Work *Jassa* [11] is a JavaScript suite for SPARQL access, which among other things provides support for client-side faceted search. For creating simple faceted search applications Jassa is, however, a much larger and more complex solution than *SPARQL Faceter*.

Sparklis [1] is an application for exploring and querying data from a SPARQL endpoint using a natural language interface[13], which also implements a faceted search. *Sparklis* focuses on data exploration and requires no prior knowledge of the data model of the target endpoint, whereas *SPARQL Faceter* aims to support development of applications tailored for a specific dataset.

Oren et al. [8] have developed a prototype for faceted navigation of arbitrary RDF data with automatic facet selection, as a server-side solution.

Future Work Plans for future development include:

1. **Semi-automatic adaptation to new datasets.** The tool could create a basic configuration for a new SPARQL endpoint, possibly based on only a class selection by the user.
2. **Customizing facet appearance.** To support customizing the appearance of the facets further , we are planning a configuration option for the user to define their own template for the facets.

Acknowledgements Kasper Apajalahti and Jouni Tuominen contributed to an earlier version of our tool. This work was supported by the Linked Open Data Science project[14] funded by the Ministry of Education and Culture in Finland.

References

1. Ferré, S.: Expressive and scalable query-based faceted search over sparql endpoints. In: The Semantic Web–ISWC 2014, pp. 438–453. Springer (2014)

[13] http://www.irisa.fr/LIS/ferre/sparklis/
[14] http:/seco.cs.aalto.fi/projects/lodsci/

2. Hearst, M., Elliott, A., English, J., Sinha, R., Swearingen, K., Lee, K.P.: Finding the flow in web site search. CACM 45(9), 42–49 (2002)
3. Hearst, M.: Design recommendations for hierarchical faceted search interfaces. In: ACM SIGIR workshop on faceted search. pp. 1–5. Seattle, WA (2006)
4. Hyvönen, E., Saarela, S., Viljanen, K.: Application of ontology-based techniques to view-based semantic search and browsing. In: The semantic web: research and applications. First European Semantic Web Symposium (ESWS 2004). pp. 92–106. Springer-Verlag (2004)
5. Hyvönen, E., Tuominen, J., Alonen, M., Mäkelä, E.: Linked Data Finland: A 7-star model and platform for publishing and re-using linked datasets. In: The Semantic Web: ESWC 2014 Satellite Events, Revised Selected Papers. pp. 226–230. Springer-Verlag (May 2014)
6. Hyvönen, E., Heino, E., Leskinen, P., Ikkala, E., Koho, M., Tamper, M., Tuominen, J., Mäkelä, E.: WarSampo data service and semantic portal for publishing linked open data about the Second World War history. In: The Semantic Web—Latest Advances and New Domains (ESWC 2016). Springer-Verlag (May 2016)
7. Koho, M., Hyvönen, E., Heino, E., Tuominen, J., Leskinen, P., Mäkelä, E.: Linked death—representing, publishing, and using Second World War death records as linked open data. In: Proceedings of the 1st Workshop on Humanities in the Semantic Web - WHiSe (at ESWC 2016) (May 2016)
8. Oren, E., Delbru, R., Decker, S.: Extending faceted navigation for RDF data. In: The Semantic Web—ISWC 2006, pp. 559–572. Springer-Verlag (2006)
9. Pollitt, A.S.: The key role of classification and indexing in view-based searching. Tech. rep., University of Huddersfield, UK (1998), http://www.ifla.org/IV/ifla63/63polst.pdf
10. Sacco, G.M.: Dynamic taxonomies: guided interactive diagnostic assistance. In: Wickramasinghe, N. (ed.) Encyclopedia of Healthcare Information Systems. Idea Group (2005)
11. Stadler, C., Westphal, P., Lehmann, J.: Jassa: a JavaScript suite for SPARQL-based faceted search. In: Proceedings of the 2014 International Conference on Developers-Volume 1268. pp. 31–36. CEUR-WS. org (2014)
12. Tunkelang, D.: Faceted search, Synthesis lectures on information concepts, retrieval, and services, vol. 1. Morgan & Claypool Publishers (2009)