SemFacet: Faceted Search over Ontology Enhanced Knowledge Graphs*

B. Cuenca Grau[1] E. Kharlamov[1] Š. Marciuška[2] D. Zheleznyakov[1] M. Arenas[3]

[1]University of Oxford [2]Microsoft Bing [3]Pontificia Universidad Catolica de Chile

Abstract. In this demo we present the SemFacet system for faceted search over ontology enhanced Knowledge Graphs (KGs) stored in RDF. SemFacet allows users to query KGs with relatively complex SPARQL queries via an intuitive Amazon-like interface. SemFacet can compute faceted interfaces over large scale RDF datasets by relying on incremental algorithms and over large ontologies by exploiting ontology projection techniques. SemFacet relies on an in-memory triple store and current implementation bundles JRDFox, Sesame, Stardog, and PAGOdA. During the demonstration the attendees can try SemFacet by exploring Yago KG.

1 Introduction

Knowledge graphs (KGs) such as Yago and Freebase have become a powerful asset for enhancing search, and are being intensively used in both academia and industry. Many existing KGs are either available as Linked Open Data, or they can be exported as RDF datasets enhanced with background knowledge in the form of an OWL 2 ontology.

Faceted search is the de facto approach for exploratory search in many online applications such as Amazon and Booking.com. Faceted search allows for querying collections of entities where users can narrow down the search results by progressively applying filters, called *facets* [6]. A facet typically consists of a predicate (e.g., 'gender' or 'occupation' when querying entities about people) and a set of possible string values (e.g., 'female' or 'research'), and entities in the collection are annotated with predicate-value pairs. During faceted search users iteratively select facet values and the entities annotated according to the selection are returned as the search result.

Faceted search in the context of RDF has received significant attention and a number of systems have been developed (see [1] for an overview of these systems). In our previous studies [1] we have developed a rigorous theoretical underpinnings for faceted search in the context of RDF-based KGs enhanced with OWL 2 ontologies: we identified well-defined fragments of SPARQL that can be naturally captured using faceted search as a query paradigm, and established

* This demo is accompanying our ISWC'16 presentation selected at the journal papers track [1]. We note that SemFacet presented here extends the one demonstrated earlier [2]: we have extended the backend with new reasoners, implemented our keyword search engine, improved the front end, and implemented incremental update of faceted interfaces. This research was supported by the Royal Society, the EPSRC projects Score!, DBOnto, MaSI^3, and ED^3 and the EU FP7 project Optique (n. 318338).

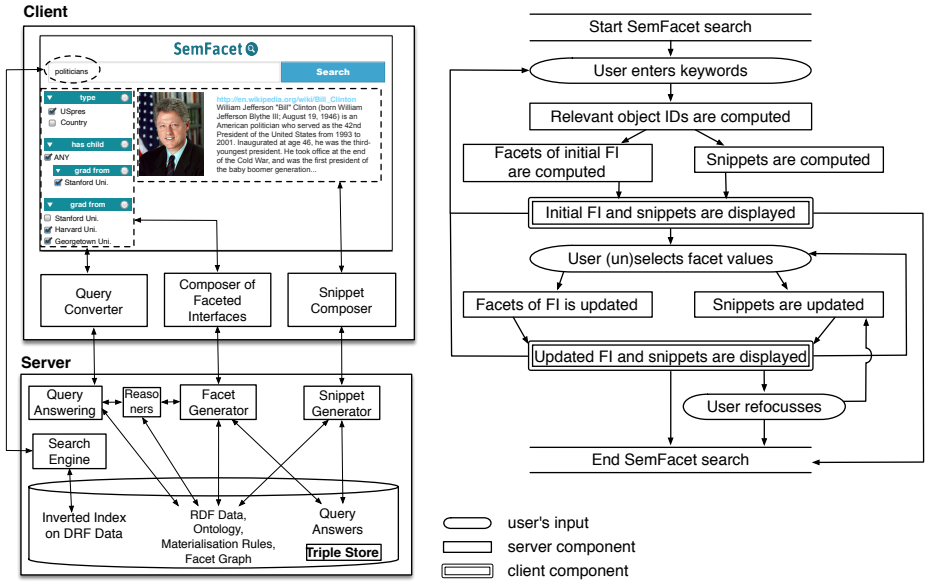

Fig. 1: Left: SemFacet workslow; Right: SemFacet architecture and screenshot

the computational complexity of answering such queries; we also studied the problem of updating faceted interfaces, which is critical for guiding users in the formulation of meaningful queries during exploratory search.

In this demonstration we present our faceted search system SemFacet that implements thiese techniques and demonstrate it on the Yago KG [5].

2 SemFacet System

SemFacet is implemented in Java and available for download under an academic license. The system can be obtained from our project website [4], where we also provide a collection of test data and detailed installation and configuration instructions. SemFacet is also available on GitHub [3].

System Architecture. SemFacet is based on a modular architecture, which is depicted in Figure 1 (Left). On the client side, SemFacet implements a GUI developed using HTML 5 consisting of three main parts: a free text search box for keywords, a hierarchically organised faceted interface, and a scrollable panel containing snippet-shaped answers. In the upper part of Figure 1 (Left) we present a screenshot of SemFacet GUI that corresponds to the following search scenario: the user is looking for politicians who are US presidents, graduated from Harward or Georgetown, and whose children graduated from Stanford. User keywords such as 'politicians' are sent by the client to the server where they are processed by the search engine. For efficiency reasons, we implemented our own simple engine based on an inverted index, and also allowed for the possibility of delegating keyword search to Lucene.

User selections in the faceted interface are compiled into a SPARQL query using the *query converter* and then sent to the back-end reasoner for evaluation. The *snippet and interface composers* receive information about facets and answers that should be displayed to the user and update the currently displayed interface and query answers. In Figure 1(Left) the answer is Bill Clinton and it is displayed

in the form of a snippet with a photograph and wiki description. The system updates the faceted interface incrementally: only the parts of the interface that are affected by users' actions are updated, which allows for a significantly faster response time. On the server side, the system relies on an *in-memory triple store* to store the inverted index, input data and ontology and other important information. The current implementation bundles JRDFox, Sesame, Stardog, PAGOdA, and HermiT. Any other in-memory triple store providing similar functionality can be seamlessly integrated with SemFacet. The *facet generator* is the back-end component responsible for constructing the interface in response to user actions, while the *query answering component* of the back-end executes the SPARQL query obtained from the query converter using the reasoning engine selected by the user.

SemFacet Workflow SemFacet's workflow is summarised in Figure 1 (Right). The user initiates the search by entering a set of keywords (e.g., 'politicians'), which are then matched to textual information associated to URIs in the data (such as labels and descriptions) resulting in an initial set of *relevant URIs*.[1] SemFacet then computes the initial interface (with no value selections) based on these relevant URIs, which constitutes the starting point for faceted navigation. The main tasks performed by SemFacet are realised in the system as follows:

- *Matching of keywords.* SemFacet exploits the values of annotation properties to determine whether a URI is relevant to a set of keywords. Intuitively, a URI u is relevant to a keyword k w.r.t. an annotation property R if the input data has a triple of the form (u, R, w), where w is a string containing k; and u is relevant to a set of keywords if at least one of them occurs in w.
- *Interface generation and update.* In order to generate and update faceted interfaces SemFacet relies on a so-called *facet graph* that consists of the input RDF data and a projection of the input ontology on a graph structure (see [1] for details). The part of this graph that corresponds to entailed RDF data is materialised offline at loading time, while the part corresponding to the ontology is computed in the online phase by querying the materialised graph. Faceted interfaces are updated in response to user actions incrementally moreover, SemFacet minimises faceted interfaces by hiding facet values whose selection leads to empty answer sets or does not affect the currently computed answer set. Finally, the current version of the system can be customised so that facet values are hierarchically arranged according to a user-specified predicate, which greatly facilitates navigation in the presence of a large number of values per facet.
- *Query generation and execution.* SemFacet compiles user selections from the faceted interface into SPARQL queries, which are then evaluated using a reasoner. Our system currently bundles several reasoning engines with different capabilities, and users can select the reasoner that is deemed more appropriate for their application at hand. Answers to SPARQL queries are typically returned by reasoners in the form of a URI. This may not be very informative for end users; hence, SemFacet also displays the annotations associated to the answer URIs and displays them in the form of a snippet.

[1] If the given set of keywords is empty, the system considers all URIs in the data as relevant.

Configuring the System. SemFacet offers a range of options for system administrators to deploy and configure the system These include *(i)* the reasoning engine of choice (JRDFox, PAGOdA, Sesame, Stardog, or HermiT); *(ii)* the annotation properties relevant for keyword search and displaying of query answers; and *(iii)* the facet that is first displayed to the user. By default, values within a facet are interpreted disjunctively; however, SemFacet provides advanced configuration capabilities for specifying which facets must be interpreted conjunctively. Additionally, the hierarchical display of facet values can also be configured by specifying the property used to construct the hierarchy (typically *rdfs*:*subClassOf* or a property capturing a partonomy relation).

3 Demonstration Scenarios

The demo attendees will experience Yago with SemFacet. They will be able to search Yago either using their own search tasks or using the tasks that we prepare for them. We now discuss the dataset that we prepare for the demonstration.

Yago comes in several slices available for download and since the current version of SemFacet relies on main memory triple stores, we took only some slices of Yago that could fit in the main memory of our machine. In particular, we took the *Taxonomy* slice, which consists of domain and range restrictions as well as subclass relations, and the *Core* slice, which contains instances of object and annotation properties. The axioms from *Taxonomy* constitute the ontology that we prepared for the demo. We refer to this ontology together with the *Taxonomy* and *Core* data slices as *FYago*. In order to generate snippets we included in both slices DBpedia abstracts, thumbnails, and links to Wikipedia articles.

FYago contains 97 million triples involving over 3 million URIs among which 55% of triples relate entities via object properties, 16% relate entities to numbers, 2% relate entities to dates, and 27% relate entities to other kinds of strings.

FYago has 89 predicate URIs which gives an upper bound to the number of facets that the user will see during faceted search. We analysed popularity of this predicates, that is how ofter the users will see them during faceted search and found out that 8 facet predicates have popularity exceeding 1 million entities and include *hasLongitude*, *hasLatitude*, and *rdf:type*, thus, a facet involving such predicate will occur in most search sessions. Then, 12 facet predicates with popularity between 100,000 and 1 million, which implies that they will occur rather often. The remaining 69 facet predicates have popularity below 100,000, and thus they will occur rarely; e.g., a facet predicate with popularity 1,000 is relevant to 1,000 entities only, and hence only to 0.025% of all data triples.

4 References

[1] M. Arenas et al. Faceted search over RDF-based knowledge graphs. In: *J. Web Sem.* 37 (2016).
[2] M. Arenas et al. SemFacet: Semantic Faceted Search over Yago. In: *Proc. of WWW (Companion Volume)*. 2014.
[3] *GitHub of SemFacet*. `https://github.com/semfacet`.
[4] *SemFacet Project Page*. `http://www.cs.ox.ac.uk/isg/tools/SemFacet/`.
[5] F. M. Suchanek et al. Yago: a core of semantic knowledge. In: *WWW*. 2007.
[6] D. Tunkelang. *Faceted Search*. Morgan & Claypool Publishers, 2009.