PepeSearch: Easy to use & easy to install semantic data search

Guillermo Vega-Gorgojo[1], Laura Slaughter[2], Martin Giese[1], Simen Heggestøyl[1], Johan Wilhelm Klüwer[3], and Arild Waaler[1]

[1] Department of Informatics, University of Oslo, Norway
{guiveg, martingi, simenheg, arild}@ifi.uio.no
[2] Oslo University Hospital, Norway laura.slaughter@gmail.com
[3] Det Norske Veritas (DNV), Høvik, Norway Johan.Wilhelm.Kluewer@dnvgl.com

Abstract. Despite the increasing availability of RDF datasets, searching and browsing semantic data is still a daunting task for mainstream users. With PepeSearch, it is easy to query an arbitrary triple store without previous knowledge of RDF/SPARQL. PepeSearch offers a form-based interface with simple and intuitive elements such as drop-down menus or sliders that are automatically mapped from the ontological structures of the target dataset. In this demonstration we will show how to set up a PepeSearch instance, how to formulate queries and how to retrieve results.

1 Introduction

An increasing number of RDF datasets is available across all domains and, as a result, many non-programmers are expressing a need for exploring these datasets. The problem is that accessing semantic data requires proficiency in SPARQL, as well as familiarity with the specific vocabularies or ontologies employed by the dataset. Alternatives to searching directly with SPARQL are mainly visual query approaches, especially graph-based query editors, e.g. QueryVOWL [1], NITELIGHT [2]. While this type of interfaces can easily exploit the graph structure of RDF and SPARQL, mainstream users are not particularly comfortable with graph visualizations [3, 4], making this approach questionable for this user group. Moreover, many common querying tasks do not require the expressivity of full graph-based querying.

We propose PepeSearch [5], a portable form-based search interface for querying semantic RDF datasets specifically aimed at helping mainstream users in their search tasks. Forms allow the user to exploit the ontology without manipulation of graph structures. Instead, the end-user employs drop-down menus, free-text entry fields, and sliders to specify classes, properties, strings, and data value ranges of their queries. This frees the user from having to invest a significant amount of time learning technical characteristics of the dataset, e.g., what an OWL class is or what ontologies are used to describe the data.

Form-based interfaces tend to be designed for specific search tasks in a single domain. User experience and design work is therefore linked to a specific context.

In contrast, PepeSearch exploits the self-describing nature of RDF and schema-level queries in SPARQL to develop a generic and portable solution that can run on any SPARQL endpoint. We allow the mainstream user to pose queries ranging from simply retrieving the members of a class, to queries joining multiple concepts and setting restrictions on datatype properties. So far, PepeSearch has been applied for use in two different contexts: government organizational data and healthcare. We will demonstrate PepeSearch at ESWC 2016: how to set up a PepeSearch instance, how to formulate queries and how to retrieve results.

2 Overview of PepeSearch

PepeSearch is an open source project under the Apache license developed at the University of Oslo[4]. It consists of the SPARQL analyzer[5], the PepeSearch component[6], and a text search engine – see Fig. 1. The provided GitHub repository also includes a screencast[7] and a live demo[8].

The analyzer is employed in a bootstrapping stage to gather information about the target data set. Through a series of generic SPARQL queries, the analyzer obtains the classes employed in the dataset, their datatype properties, and the connections to other classes through an object property or through a subclass relation. The result is a data schema in the JSON format.

The obtained data schema can then be used to configure a PepeSearch instance. The query builder component is in charge of preparing a suitable view for querying the dataset. For an arbitrary RDF class, a form block is created, in which datatype properties are mapped to widget elements. In order to support multi-class queries, a collapsible form block is included for each RDF class that is connected with an object property to the selected class – see Fig. 2(a) for an example. The results viewer element is in charge of sending the query to the SPARQL endpoint and presenting the results in a tabular representation – see Fig. 2(b). Browsing is supported through the instance viewer that obtains all the data about a particular individual with links to other connected instances – see Fig. 2(c).

The text search engine is an optional component that allows dynamic term suggestions during query specification. This is employed to provide autocomplete capabilities for the text fields of a class, e.g. to suggest names such as “Martin” or “Maria” after typing “mar” in a name textbox.

3 Hands on with PepeSearch

To illustrate the operation of PepeSearch, we will employ a sample dataset containing health records of fictitious patients. Anonymized patient data has been

[4] `http://www.uio.no/`
[5] `https://github.com/simenheg/sparql-endpoint-analyzer`
[6] `https://github.com/guiveg/pepesearch`
[7] `http://folk.uio.no/simenheg/pepesearch.webm`
[8] `http://sws.ifi.uio.no/project/semicolon/search/`

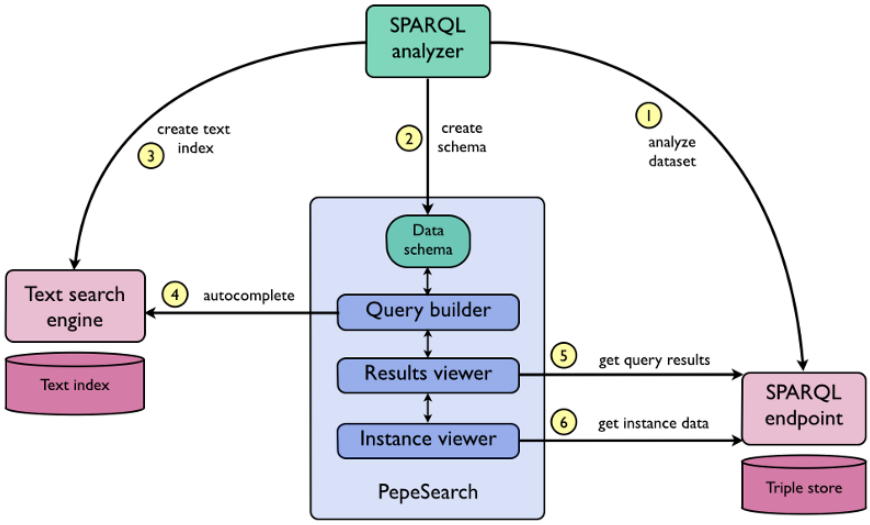

Fig. 1. Logical architecture of PepeSearch.

provided by our hospital project partner in the form of tables from a widely used hospital records application. It describes health care processes, with associated diagnoses and medical personnel in various roles, supported by a body of code lists. This data is mapped into RDF according to an ontology with three main parts: (i) excerpts from the Disease Ontology[9] to cover the medical conditions that appear in the data, (ii) the Information Artifact Ontology[10] for documents, and (iii) local extensions for measurements of vital signs and for a part/whole hierarchy of health care processes. Upper classes and relations are provided by the OBO Relations Ontology[11].

As an example, we show how to obtain a set of patients between 30–50 yrs of age that have suffered from an intestinal disease. Use of semantic technologies for cohort identification has been proposed [6], and is an important application area. We first run the SPARQL analyzer to generate the data schema out of the dataset structure with all the classes, properties and value types. PepeSearch can then be used to fulfill the aforementioned information need in this way:

1. PepeSearch presents a list of the top classes available in the dataset.
2. We select the concept "human being".
3. PepeSearch presents a form block for the "human being" class and a list of collapsibles corresponding to classes directly connected to "human being" in the dataset, e.g. "diagnosis" or "health care encounter".
4. We set the restrictions required for this search task: in the "human being" class we select "patient" as a more specific type; we use the age slider to set

[9] `http://disease-ontology.org/`
[10] `https://github.com/information-artifact-ontology/IAO/`
[11] `https://github.com/oborel/obo-relations`

the appropriate range; and we select the "intestinal disease" after expanding the "disposition" collapsible. A snapshot of this query is shown in Fig. 2(a).
5. We push the "Get results" button at the top right corner of the search interface.
6. Behind the scenes, PepeSearch generates a SPARQL query from the form that is sent to the SPARQL endpoint.
7. With the response, PepeSearch prepares a tabular representation of the results (see Fig. 2(b)).
8. We can navigate through the results by following the links, e.g. Fig. 2(c) shows the information of one of the patients found.

4 Conclusions

PepeSearch is a portable form-based interface for searching semantic data sets devised for mainstream users. In this demonstration we will present the different components of PepeSearch. We will use the SPARQL analyzer to gather the data schema of several triple stores, and we will then use PepeSearch to formulate queries and retrieve results.

Acknowledgements

This work has been partially funded by the Norwegian Research Council through the HealthInsight project (NFR 247784/O70), and the European Commission through the Optique (FP7 GA 318338), and BYTE (FP7 GA 619551) projects.

References

1. Haag, F., Lohmann, S., Siek, S., Ertl, T.: QueryVOWL – Visual Composition of SPARQL Queries. In: Proceedings of the 12th European Semantic Web Conference (ESWC2015), Portoroz, Slovenia (2015)
2. Russell, A., Smart, P.R., Braines, D., Shadbolt, N.R.: Nitelight: A graphical tool for semantic query construction. In: Semantic Web User Interaction Workshop (SWUI 2008), Florence, Italy (2008)
3. Viégas, F.B., Donath, J.: Social network visualization: Can we go beyond the graph? In: Proceedings of the Computer Supported Cooperative Work (CSCW'04), Workshop on Social Networks. Volume 4., Banff, Canada (2004) 6–10
4. Elbedweihy, K., Wrigley, S.N., Ciravegna, F.: Evaluating semantic search query approaches with expert and casual users. In: Proceedings of the 11th International Conference on The Semantic Web (ISWC 2012), Boston, MA, USA, Springer-Verlag (2012) 274–286
5. Vega-Gorgojo, G., Giese, M., Heggestøyl, S., Soylu, A., Waaler, A.: PepeSearch: Semantic data for the masses. PLOS ONE (2016) URL: `http://dx.doi.org/10.1371/journal.pone.0151573`.
6. Pathak, J., Kiefer, R.C., Chute, C.G.: Using semantic web technologies for cohort identification from electronic health records for clinical research. AMIA Summits Transl Sci Proc **2012** (2012) 10–19

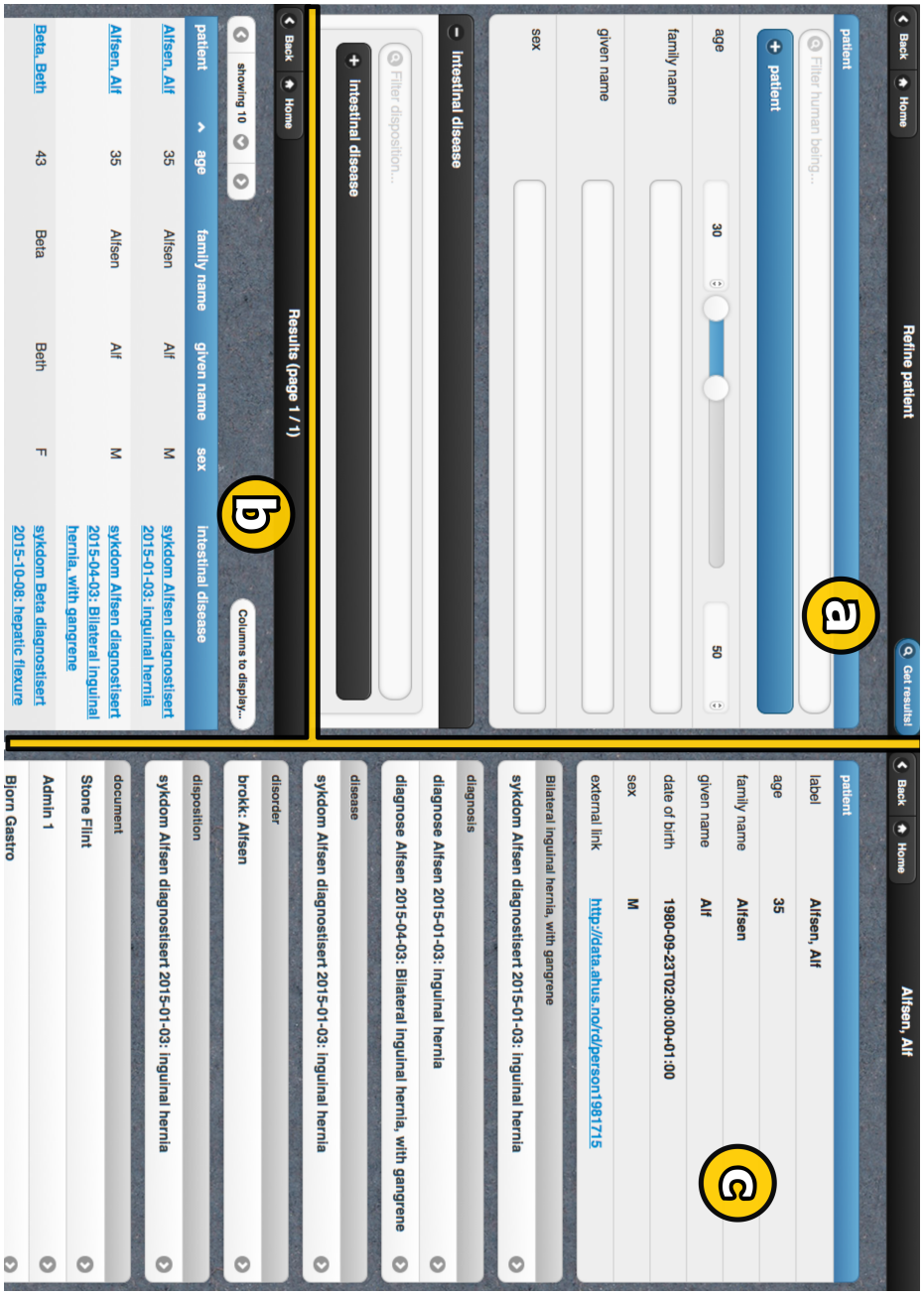

Fig. 2. Snapshots of PepeSearch.