An interactive visualisation for RDF data

Fernando Florenzano[1 2], Denis Parra[1], Juan Reutter[1 2], and Freddie Venegas[1]

[1] Pontificia Universidad Católica de Chile
[2] Center for Semantic Web research, CL

Abstract. We demonstrate a visualisation aimed at facilitating SPARQL-fluent users to produce queries over a dataset they are not familiar with. This visualisation consists of a labelled graph whose nodes are the different types of entities in the RDF dataset, and where two types are related if entities of these types appear related in the RDF dataset. To avoid a visual overload when the number of types in a dataset is too big, the graph groups together all types that are subclass of a more general type, and users are given the option of navigating through this hierarchy of types, dividing type nodes into subtypes as they see fit. We illustrate our visualisation using the Linked Movie Database dataset, and offer as well the visualisation of DBpedia.

1 Introduction

Today one can reasonably state that querying an endpoint is an easy task, assuming of course that one is familiar with the SPARQL language and with the structure of the dataset made available by the endpoint. Unfortunately, this is a pretty strong assumption: even if the endpoint is up and running, and even if the user is an expert in SPARQL, the task of producing a query that extracts the desired information may end up demanding more resources than those needed to actually compute the query. The problem is the unstructured nature of RDF: as there is no real notion of schema, knowing what and how the RDF data is stored is not an easy task. In contrast with relational databases, where users can directly consult the schema information for the names and attributes of tables, with RDF data one has almost no alternative but to understand the data by issuing several probe queries. This behaviour has been confirmed when analysing query logs of different endpoints [1, 9].

As an example of the challenges we face, consider and endpoint for *Linked Movie Database* (LinkedMDB [3]), a database storing information about movies: who starred them, who directed them, when and where were they shot, etc. Consider now a SPARQL expert trying to obtain the names of all directors that have also acted in one of their movies. The landing page of our LinkedMDB endpoint of choice would probably look as a white canvas, perhaps offering as well a few example queries, so, unless we are lucky and hit an example, there are almost no clues on how to start answering the desired query. In our case, to answer this query we would need information about how are actors and directors

stored in the database, how is the connection between actors, directors and movies, and how to recognise when a director and an actor are the same person.

It has been noted that users new to RDF datasets would benefit tremendously from a graphical interface that explicitly gives them the information they need to start producing meaningful SPARQL patterns (see e.g. [2, 6, 5, 8]). But if we want ordinary endpoint users to take advantage of our interface, we need a lightweight system that can be added onto endpoints without much computational overload and that can scale for arbitrary large datasets.

Related work. To our best knowledge, none of the available visualisations offer a summarised graph with the ability to navigate through the graph using hierarchy of types. For example, systems such as LODSight [2], Sparqture [8], those working with VOWL [7] or that of [6] provide rather static visualisations, and systems such as LODpeas [4] provide interactive visualisations but only at an entity level.

2 Overview of the System

Our main visualisation is a labelled, undirected graph, where the nodes are the types of the dataset, and where there is a p-labeled edge between nodes t_1 and t_2 if and only if there are entities u_1 and u_2 such that u_1 is of type t_1, u_2 is of type t_2 and the dataset contains the triple $\{u_1\ \ p\ \ u_2\}$. That is, two types are related to each other if there are entities of these types connected by a triple in the dataset. We complement the main graph with two extra panels: a panel on the left side that allows the users to obtain additional information of the semantic graph as a whole and a panel on the right with information about the specific node or edge selected on the main pane. Figure 1 presents a screenshot of our visualisation applied to the LinkedMDB dataset. The central pane is the graph, the left-side panel displays the top 10 types ranked according to the number of total entities of this type, and the right-side panel shows specific information about the node selected by the user (in this case *performance*). In what follows we give a brief description of the main features of our visualisation.

Hierarchical navigation of types. Most of the type assignment in more complex RDF datasets is hierarchical. We take advantage of the forest-like shape of the RDF type hierarchy and include in our visualisation the ability to *slice* the type graph in different levels of this hierarchy. Figure 2(a) describes the same visualisation of LinkedMDB, but where some nodes are hidden and the emphasis is on the Person node. In LinkedMDB there are several types which are a subclass of Person, and users interested in one of them can drill down on this node, subdividing Person into its subclasses, as shown in Figure 2(b).

Navigation and summarisation of relations. It is not strange to find different relations amongst entities of the same type in an RDF dataset. For example, there are several relations between persons and films in LinkedMDB. As we did with nodes, we aggregate all relationship into one big set, and only show the more fine-grained relationships whenever they are requested by the user. This again prevents information overload, and allow to search only those relations that are interesting to the user.

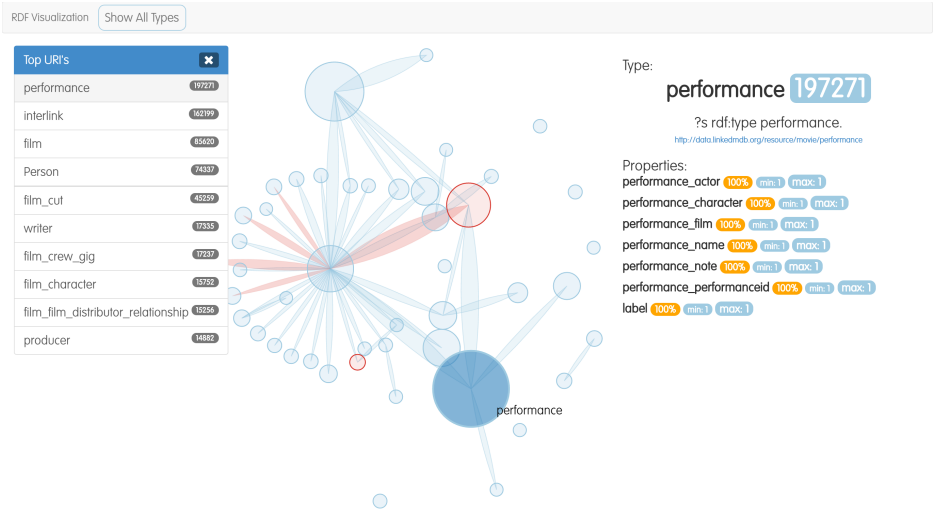
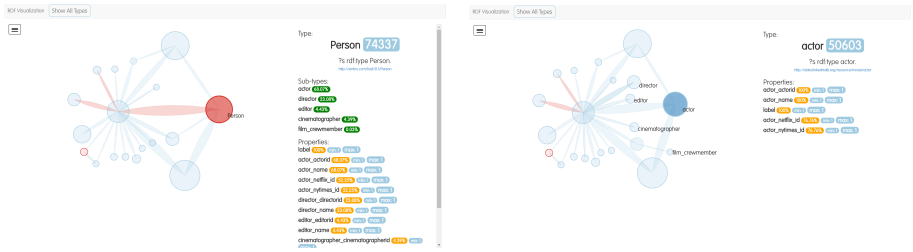


Fig. 1. The top-level visualisation of LinkedMDB's dataset



(a) Graph with Person node (b) Result of dividing the Person node

Fig. 2. Visualisation of LinkedMDB when the Person node is subdivided into actor, director, editor, cinematographer and film_crewmember.

Summarization of attributes. An attribute of a certain entity u is a property that relates u with a string or value, such as performance_actor, performance_character, etc., in LinkedMDB, according to Figure 1. Attributes are vital in asking questions such as *What is the name of...?* or *In which year did...?*. However, not all entities of the same type have the same attributes, and it is difficult to show all the attributes of all types in the graph at the same time. For this reason, when users hover over or select a particular node in the graph, the right-side panel displays all the information about the attributes of entities of this type. Additionally, a percentage is shown that corresponds to the percentage of entities of the chosen type that posses this attribute.

System architecture. The force layout was constructing using the D3 library. To show the visualisation of RDF datasets we first preprocess them, using SPARQL, to generate a JSON file that contains only the needed information

about the datasets. This file is mounted on a server, and clients using the interface may send server requests to query this file. This keeps the visualisation quite low on computational resources, the only computationally expensive part is the pre-computation, but this can be done in regular intervals.

Scalability and limitations. Of course, this way of visualising RDF data can only scale as long as one can find a strict hierarchy of types: a *flat* RDF graph in which no type is a subclass of another type will be visualised just as any other force graph layout. Furthermore, even tough the system aggregates information into types, and therefore scales perfectly well over increasingly large dataset, it does suffer from scalability issues when the ontology is too large: running the visualisation over YAGO or DBPedia datasets (which have millions of types) does require computational powers beyond a personal computer.

3 Demonstration Overview

The demonstration will showcase the visualisations shown in our live server at http://jreutter.sitios.ing.uc.cl/VisualRDF.html. We start with LinlkedMDB, explaining the basic navigation options of our example, and then we show the scalability of the system by showing a portion of the YAGO database with more than a million types. We will also provide live access to endpoints of these datasets, and ask users to produce queries using only the visualisation as information for the dataset. The aim of the demo is to showcase the usefulness of these types of visualisations to produce SPARQL queries, and how navigating through type hierarchies is an important feature of these tools.

Acknowledgements. Work funded by the Millennium Nucleus Center for Semantic Web Research under Grant NC120004.

References

1. C. Buil-Aranda, M. Ugarte, M. Arenas, and M. Dumontier. A preliminary investigation into SPARQL query complexity and federation in Bio2RDF. In *AMW*, page 196, 2015.
2. M. Dudáš, V. Svátek, and J. Mynarz. Dataset summary visualization with LODSight. In *ESWC*, pages 36–40. Springer, 2015.
3. O. Hassanzadeh and M. P. Consens. Linked movie data base. In *LDOW*, 2009.
4. A. Hogan, E. Munoz, and J. Umbrich. LODPeas: Like peas in a LOD (cloud). *Proceedings of the Billion Triple Challenge*, 2012.
5. S. Khatchadourian and M. P. Consens. ExpLOD: summary-based exploration of interlinking and RDF usage in the linked open data cloud. In *ESWC*, pages 272–287. Springer, 2010.
6. S. Kinsella, U. Bojars, A. Harth, J. G. Breslin, and S. Decker. An interactive map of semantic web ontology usage. In *2008 12th ICIV*, pages 179–184. IEEE, 2008.
7. S. Lohmann, S. Negru, F. Haag, and T. Ertl. Visualizing ontologies with vowl. *Semantic Web*, (Preprint):1–21, 2016.
8. F. Maali. SPARQture: a more welcoming entry to SPARQL endpoints. In *IESD*, pages 78–82. CEUR-WS. org, 2014.
9. M. Saleem, M. I. Ali, A. Hogan, Q. Mehmood, and A.-C. N. Ngomo. LSQ: The linked SPARQL queries dataset. In *ISWC*, pages 261–269. Springer, 2015.