Browsing Linked Data with MyView

Gong Cheng, Huiyao Wu, Saisai Gong, Hang Zhang, and Yuzhong Qu

State Key Laboratory for Novel Software Technology, Nanjing University,
Nanjing 210093, China
{petercheng456,sparklewhy,saisaigong,hzhang.nju}@gmail.com
yzqu@nju.edu.cn

Abstract. Compared with the hypertext Web, Linked Data can satisfy more precise information needs, but presently still lacks tool support for citizen users. In this demonstration, we introduce a personalizable Linked Data browser called MyView, which enables users to query Linked Data by navigation (such as link traversal and filtering) from one entity collection to another. With MyView, users can reuse their past queries in various ways, including categorizing favored links as different views, assembling complex links with existing ones, and revisiting past queries via history and bookmark mechanisms. As an intelligent system, MyView evaluates queries in a logic programming fashion which supports reasoning, and its implementation features several strategies for dealing with the distributed, open, large-scale Web environment. It also generates explainable answers that carry provenance information, and supports source control. Finally, it interacts with users to resolve entity coreference for discovering more sources and reducing redundancy.

Keywords: Linked Data browser, navigational query model, personalization, query reuse, rule-based query answering

1 Introduction

The Semantic Web and Linked Data use RDF as a common structured format for publishing data on the Web. Data is further enriched by ontologies with well-defined semantics. Thus, applications have a good chance of reusing, integrating and reasoning with data for providing more effective services such as answering formal queries which characterize precise information needs. Considering the difficulty that citizen users have in constructing formal queries, different supporting modes of interaction [2, 4, 5] have been developed. However, they actually deem Linked Data a local database of which the whole schema and data are available for their approaches, but the fact is that users are usually aware of only a small portion such as an entity URI or a collection of entities to start with. Then, from these entry points, the Web needs to be explored to formulate queries and retrieve corresponding data. Following this direction, Linked Data browsers such as Tabulator [1] have been developed. When these tools and subsequent research (e.g. [3]) really begin to look at the "Web" aspect of Linked Data, facilities ranging from fitting citizen users' skill to allowing user-directed personalization have not been addressed extensively.

In this demonstration, we will present a new personalizable and intelligent Linked Data browser, called MyView,[1] which is fully targeted on citizen users. We will briefly describe its main features in the next section.

2 Features of MyView

2.1 Navigational Query Formulation

We intend to support complex formal queries in MyView. Because direct construction of formal queries is too difficult for citizen users to perform, we enable them to easily achieve it via navigation between entities in a way that is quite similar to navigation between webpages on the hypertext Web. In particular, in line with [2], first-class citizens in MyView are not individual entities but could be collections of entities. That is, users can navigate from one collection to another via comforting interaction, whereas essentially they construct and submit formal queries comprising link traversal and filtering. To be specific, from an entity collection C, three types of navigations are supported:

Link Traversal returns an entity collection comprising all the resources reachable from at least one entity in C by following a specified type of link.
Type Filtering returns a subset of C retaining all that are of a given type.
Link Filtering returns a subset of C retaining all that, by following a specified type of link, will reach at least one resource in a given collection of resources.

Because the result of every navigation is again an entity collection, for a given one to start with, we could successively perform a series of (different) navigations, which results in a complex formal query. This navigational query model forms the basis of MyView. Given an entity collection and available data, users could interact with MyView to perform any such navigation that would not lead to empty result, by simply clicking and making selections.

2.2 Query Reuse

MyView provides three mechanisms for (partially) reusing queries previously constructed, to establish a personalized user experience, thereby improving the efficiency in interaction.

View Customization. When browsing an entity collection in MyView, initially all their links (and linked resources) that can be found in available data will be presented. To avoid overloading users with too many links, MyView allows to freely organize links into views. A view is conceived as a layout of a set of specified types of links. Users can include any links in a view, rename each link, and choose a layout of them. Each view is bound to a class. When browsing an entity collection, all and only the views bound to their common classes (including inferred superclasses) will be shown.

[1] MyView is downloadable from `http://ws.nju.edu.cn/explorer/myview/`.

Link Customization. In MyView, a link could be far beyond a property explicitly defined in an ontology. We enable users to create complex links by using existing ones. Firstly, we allow to define a link as the inverse of another link, i.e. following it in a backward direction. Secondly, several specific types of links can be generalized to a single one, e.g. merging "has-father" and "has-mother" into "has-parent". Thirdly, we allow to create a shortcut link to represent a chain of links, e.g. "has-uncle" as a concatenation of "has-aunt" and "has-husband". Once created, such custom links can also be used for link traversal and filtering, and being included in a view.

Query Revisit. MyView provides a bundle of means of query revisit, i.e. rapid resubmission of a past query. Firstly, MyView records all the navigation actions performed by the user and organizes them into trees. The user can resubmit a past query by clicking the corresponding tree node, and can create a new branch from it, i.e. constructing a new query by reusing (part of) a past query. Secondly, MyView inherits stack-based back/forward buttons as well as browsing history organized by the time of visit from major Web browsers. Thirdly, users could store an entity URI as a bookmark. In the more general case of an entity collection resulting from a query, we allow to store the query itself rather than the computed results so that in the future it could be reevaluated on the fly to obtain the latest results.

2.3 Query Evaluation over Linked Data

In MyView, ontologies and RDF data are transformed into Datalog programs in a way similar to Description Logic Programs. Navigational queries are represented as goals to Datalog programs, and are evaluated by an embedded rule engine. Thereby, reasoning is inherently supported. MyView starts with zero knowledge; when a query arrives, MyView collects relevant rules and facts from the Web on the fly by dereferencing each newly met URI. However, our rule engine never waits for URI dereference, which may take a long time, but only rests on all the rules and facts having been downloaded so far. Since new sources arrive continually, MyView reevaluates the query periodically against updated local cache of rules and facts, and notifies users of changes.

2.4 Query Result Explanation

MyView not only presents but also explains answers. During evaluation, the provenance of each fact is recorded. For a fact directly retrieved from the Web, its provenance is simply the URI of its source, i.e. an RDF document; for an inferred fact, we record the rule and the facts that collectively introduce it. Then, the complete explanation of a fact can be represented as a proof tree. To better serve citizen users, proof trees are serialized to text in breadth-first order. Besides, when users observe low-quality information in proofs, they are allowed to prevent MyView from retrieving rules and facts from specified sources. Such blocking operations can be carried out on RDF document level or host level.

2.5 Enriching Query Evaluation with Entity Coreference Resolution

Given Web an open environment, different applications may describe the same real-world entity but use different identifiers, i.e. URIs. It causes difficulties in data integration, and may also raise semantic redundancy in query results. Considering the low-quality Web data, we address this problem of entity coreference by employing both machine and human computation. Rather than performing fully automatic coreference resolution, MyView basically leverages OWL semantics (e.g. `owl:sameAs` and inverse-functional properties) to detect coreference candidates, and then users take their responsibilities to judge whether each candidate should be accepted or not. To assist users in making judgments, each candidate is accompanied by an evidence showing those `owl:sameAs` and inverse-functional properties that introduce the candidate. In addition, judgements will be employed as feedback to improve the underlying algorithm.

3 Conclusions and Future Work

MyView is a client-side browser for querying Linked Data. When featuring a navigational mode of interaction for citizen users, it is characterized by the powerful personalization capability including view and link customization, and query revisit. It also exhibits considerable intelligence in the sense of rule-based reasoning as well as result explanation. In future, we will extend MyView to look up external indexes for acquiring more data sources for query evaluation, and realize view sharing and collaborative entity coreference resolution.

Acknowledgments. This work was supported by the NSFC under Grant 61003018. We thank all the students that participated in the development.

References

1. Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., Sheets, D.: Tabulator: Exploring and Analyzing Linked Data on the Semantic Web. In: 3rd International Semantic Web User Interaction Workshop. (2006)
2. Harth, A.: VisiNav: A System for Visual Search and Navigation on Web Data. J. Web Semant. 8(4), 348–354 (2010)
3. Hartig, O., Bizer, C., Freytag, J.-C.: Executing SPARQL Queries over the Web of Linked Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 293–309. Springer, Heidelberg (2009)
4. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data. In: 2009 IEEE International Conference on Data Engineering, pp. 405–416. IEEE Computer Society, Washington, DC (2009)
5. Zenz, G., Zhou, X., Minack, E., Siberski, W., Nejdl, W.: From Keywords to Semantic Queries — Incremental Query Construction on the Semantic Web. J. Web Semant. 7(3), 166–176 (2009)