SView: Sharing Views for Browsing Linked Data

Gong Cheng, Saisai Gong, Xiangqian Li, Qingxia Liu, Liang Zheng, Jidong Jiang, Wei Hu, and Yuzhong Qu

State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, P.R. China
gcheng@nju.edu.cn, saisaigong@gmail.com, skyline0623@gmail.com, qxliu@smail.nju.edu.cn, zhengliang@smail.nju.edu.cn, jdkjiang@gmail.com, whu@nju.edu.cn, yzqu@nju.edu.cn

Abstract. We submit to the Open Track a new system for browsing Linked Data, called SView. The main philosophy of this system is to decompose the work on producing and consuming Linked Data into fine-grained tasks and distribute them to different people, all of whom are given incentives for participating in. Specifically, it enables general Web users to conveniently browse entities and interact with Linked Data through powerful views and navigators shared by the developer community. It also encourages users to participate in data fusion in a pay-as-you-go fashion.

Keywords: Linked Data browser, multi-view presentation, pay-as-you-go data fusion, semantic navigation.

1 Introduction

The volume of Linked Data grows rapidly. Browsers such as Disco [2] and Tabulator [1] have been developed to enable general Web users to directly meet and consume Linked Data. Following this line of work, we are developing SView,[1] a new system for browsing Linked Data. In our vision, SView is designed as an open platform that receives "plug-ins" (e.g. views) from the developer community, and provides them to users to create an environment where users can conveniently browse and interact with Linked Data through these shared views. SView is distinguished from existing systems by the following features.

- SView receives both generic and domain-specific **views** developed and submitted by the developer community, and provides them to users to browse and interact with Linked Data. Views can also invoke external services and data.
- SView supports various forms of **navigators** for navigation on Linked Data, such as pivoting between (sets of) entities and filtering. Navigators can carry complex semantics beyond a single property.

[1] http://ws.nju.edu.cn/sview.

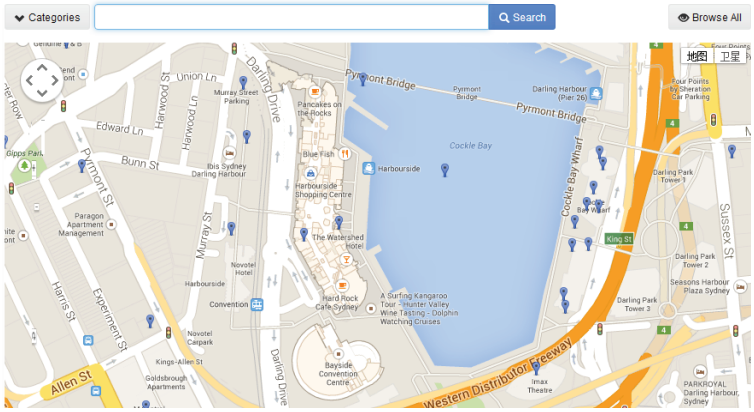

Fig. 1. A map view with POIs.

– SView conducts **pay-as-you-go data fusion**. Users are given incentives for collaboratively verifying candidate coreferent URIs computed by the system, and the process is facilitated by comparative entity summaries.

The **main philosophy** of SView is to create an ecosystem, where the work on producing and consuming Linked Data is decomposed into fine-grained tasks and distributed to different people. Specifically, data publishers provide data; view developers focus on the presentation of data; navigator developers focus on the exploration of data; and finally, users can conveniently consume data, and help fuse data. More importantly, all of them are given incentives for participating in.

The remainder of the paper is structured as follows. Section 2 and 3 introduce views and navigators, respectively. Section 4 presents the incentives given to developers for contributing. Section 5 describes our design of pay-as-you-go data fusion. Section 6 reports a user study. Section 7 concludes the paper with a discussion of future work.

2 Browsing and Interacting with Views

RDF data is essentially a set of RDF triples, or an RDF graph. However, data is not necessarily presented to the user as it is represented in the computer [4]. To focus on what users want to do with the data as information, SView supports presenting data in multiple domain-specific views, from which a user is free to choose the one that she believes is the best for solving her task. For instance, Fig. 1 shows a map view of Sydney with a number of points of interests (POIs). Some other systems like Tabulator [1] also support multiple views. The features that distinguish SView from existing systems are as follows.

Sharing views. It is difficult, if not impossible, for a single organization to exhaustively develop views for every application domain. Instead, SView is

designed as an open platform that receives views developed and submitted by the developer community, and provides them to users. By sharing views, repeated development can be reduced since one view may fit multiple data sources. Though to date, SView has received only a limited number of views, the reader is reminded of the openness of the system. It is possible to attract submissions because developers in different communities are given different incentives for developing and submitting views, which will be discussed in Sect. 4.

Invoking external services and data. In SView, a view can go beyond merely showing the data returned by the target Linked Data server. It can also invoke external services and integrate data provided by other data publishers. For instance, the map view in Fig. 1 employs Google Maps API to embed a map in the view. In particular, it is worth noting that the POIs on the map are found by executing a SPARQL query against the LinkedGeoData SPARQL endpoint with latitude, longitude, and (optional) keyword and type constraints. Users can filter POIs by adjusting the map's center and zoom levels, and optionally providing a keyword description and specifying types. All these constraints will be combined into a SPARQL query.

Connection with multimedia and the social Web. Via views, SView can easily connect Linked Data with multimedia and social data on the Web. For instance, an image view has been provided to display images about the present entity by retrieving images from Google Images by using the name of the entity as a keyword query, and also by invoking the Flickr API to search for photos shared by Flickr users by using the name of the entity as a tag constraint. The reader is encouraged to creatively contribute views that integrate other types of multimedia data and social networking services.

Personalized and dynamic view listing. The list of views to show in SView can be personalized. A user is free to choose the views to "install". She can also change the order of the installed views. When a large number of views are available in the future, we will devise a way to automatically make view recommendations to users based on view usage and ratings. Besides, not all the installed views will be shown when browsing every entity. Rather, views are triggered by the type or URI pattern of the entity, or by the existence of particular properties in its RDF description.

In particular, SView has a built-in view called 'Home' for all entities. In this generic view, the original RDF description of the present entity is presented in tabular form, and is grouped by property. The selection of properties and their order can be personalized.

3 Pivoting and Filtering with Navigators

For Linked Data browsers, another core function to serve is navigation, or, more technically, refocus. SView supports navigation from one entity to another. For instance, by clicking a POI in the map view in Fig. 1, the user will be refocused

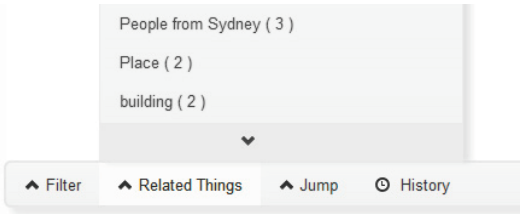

Fig. 2. Four navigators: one for faceted filtering, two for pivoting, and one for going back to historical entities.

on this new entity to browse. In accordance with recent systems like VisiNav [3], SView also allows for set-oriented navigation, i.e., navigating from one set of entities to another, either by simultaneously following a set of links labeled with the same property (called pivoting) or by filtering by a specific property value (called faceted filtering). However, SView is distinguished from existing systems by the following features.

Various forms. A navigator can be embedded in a view. For instance, when browsing the map view of Sydney in Fig. 1, the user can choose to refocus on the set of all POIs in the current field of view by clicking the 'Browse All' button. A navigator can also be independent of specific views. All such navigators will be organized into a menu. For instance, Fig. 2 shows several generic navigators that support traditional pivoting and faceted filtering. Similar to view sharing, we also plan to allow for sharing navigators. When a large number of independent navigators are available in the future, we will consider issues like personalization and recommendation.

Complex semantics. In SView, a navigator can go beyond a single property. For instance, one independent, set-oriented navigator can refocus on the people that have participated in any event in the present city. What underlies it is a complex structured query. Another example has been embedded in the map view in Fig. 1. Actually, clicking each POI on the map is conceived of as invoking an entity-to-entity navigator. In particular, though the semantic relationship between Sydney and a specific POI is not explicitly specified in terms of RDF properties, navigation can still happen.

History and bookmark. To support the "back" button functionality, SView records the history of navigation. As shown in Fig. 2, users can go back to historical entities. Besides, if the user favors an entity, she can add it to her bookmark to conveniently revisit in the future.

4 Incentives for Developing Views

The success of SView's ecosystem relies on continuous contribution of views (and navigators). In the following, we will show how several developer communities

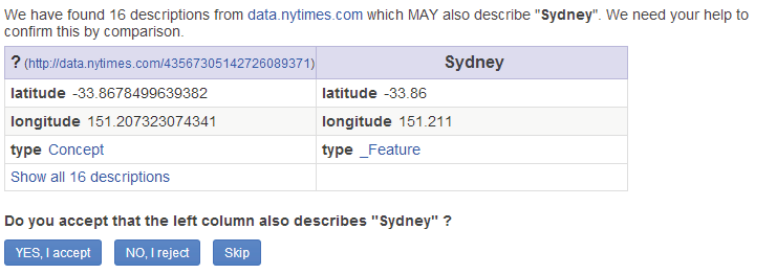

Fig. 3. Verifying candidate coreferent URIs based on a comparative summary.

will be self-motivated to contribute. More communities are to be identified in the future.

Website owners. SView is designed as an open platform, similar to those popular mobile software distribution platforms. Owners of existing websites will not ignore this emerging platform, but rather, will create views that can help to promote their websites, and in particular, can increase the traffic of their websites. For instance, online travel agencies will contribute competitive flight views in order to direct potential consumers toward their own booking services.

Independent developers. Independent developers will also actively contribute views because once their works become popular, it will not be difficult to find a way to make a profit given a large user base. For instance, developers can make a profit by embedding, in a popular flight view they developed, advertisements for relevant services such as booking.

Data publishers. If data publishers have incentives for publishing Linked Data, they may also be willing to create views that can best showcase their data, as an effective way to promote the data, to demonstrate its value and to reach out a broader range of audiences.

5 Pay-as-you-go Data Fusion

Multiple data on the Web representing the same real-world object are to be fused. However, it has been generally accepted that the management of Web-scale semantic heterogeneity requires human involvement. SView solves data fusion in a pay-as-you-go fashion [5]. Apart from explicit entity coreference such as those obtained from `owl:sameAs` links, users are invited to decide whether to accept candidate coreferent URIs computed by an automatic entity coreference algorithm. Our design is characterized by the following features.

Incentives for user involvement. Users are given incentives for participating in because we let them know that if candidate coreferent URIs are accepted, more data (and thus more views) about the present entity will be provided.

Verifying candidate coreferent URIs based on comparative summaries. To help users effectively and efficiently give feedback on candidate coreferent URIs, SView automatically computes a comparative summary of the two descriptions of entities to be compared. As shown in Fig. 3, such a summary consists of a limited number of paired properties that are believed to best help indicate the equivalence of or difference between the two entities.

Personalization and collaboration. Coreferent URIs accepted by a user are recorded to be applied in her future browsing. Meanwhile, coreferent URIs accepted by different users will be synthesized, and when conflicts occur, a consensus will be reached based on a voting strategy. As a result, each user can enjoy not only the coreferent URIs accepted by herself but also those contributed by other users. Further, all the results of verification will be used as training examples by our learning-based entity coreference algorithm.

Selecting candidate coreferent URIs. The selection of candidate coreferent URIs to be verified by a user considers two sides. On one hand, we prefer those candidate coreferent URIs that are very likely to be accepted and, once accepted, a large amount of data will be introduced, thereby maximizing the user's benefit. On the other hand, candidate coreferent URIs will be preferred if the improvement of the system's ability to identify entity coreference is maximized after knowing the result of verifying them.

6 Preliminary User Study

We invited ten graduate students having a background in computer science to participate in a preliminary user study. Firstly, participants were given a tutorial about SView, which provided five views and four generic, independent navigators at the time of experimentation. Then, participants were given 15–30 minutes to use SView to browse an author of an arbitrary ISWC paper (retrieved from the Semantic Web Dog Food Corpus), and to browse an arbitrary city (retrieved from DBpedia). Finally, they were invited to fill out a System Usability Scale (SUS) questionnaire for usability testing, and to give comments via interview.

The average SUS score is 67.5. In general, participants found the various functions in SView were well integrated, and felt very confident using the system. Among the comments, eight participants enjoyed multiple views and integrated information, and four found it convenient to use various navigators. However, all the participants complained about the speed of loading Linked Data, in particular when exploring a set of entities. Three expected to have more views.

7 Conclusions and Future Work

SView is a Web-based system for browsing Linked Data. It enables general Web users to conveniently browse entities and interact with Linked Data through shared views and navigators. A number of generic views and navigators have been built in the system to guarantee that any Linked Data is browsable. Whereas several domain-specific views and navigators have also been provided to illustrate

the potential of the system, more are expected to be plugged in by the developer community to improve user experience when browsing.

Currently, we are actively working to open the platform. We will provide an SDK for developers as soon as possible. Performance optimization is also a future work.

Acknowledgments. This work was supported in part by the NSFC under Grant 61223003, 61170068, 61100040, and 61370019, in part by the SSFC under Grant 11AZD121, and in part by the JSNSF under Grant BK2012723 and BK2011189. The authors would like to thank Yanan Zhang for her contribution to the experiments.

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 Int'l Semantic Web User Interaction Workshop. (2006)
2. Bizer, C., Gauß, T.: Disco - Hyperdata Browser, `http://sites.wiwiss.fu-berlin.de/suhl/bizer/ng4j/disco/`
3. Harth, A.: VisiNav: A System for Visual Search and Navigation on Web Data. J. Web Semant. 8(4), 348–354 (2010)
4. Karger, D., schraefel, m.c.: The Pathetic Fallacy of RDF. In: 3rd Int'l Semantic Web User Interaction Workshop. (2006)
5. Madhavan, J., Jeffery, S.R., Cohen, S., Dong, X.L., Ko, D., Yu, C., Halevy, A.: Web-scale Data Integration: You Can Only Afford to Pay As You Go. In: 3rd Biennial Conference on Innovative Data Systems Research, pp. 342–350. (2007)

Appendix: Evaluation Requirements

Table 1. Minimal requirements

	Requirement	How we have addressed
1	The application has to be an end-user application.	**Yes**. SView is designed to help general Web users browse and interact with Linked Data.
2	The information sources used should be under diverse ownership or control, be heterogeneous, and contain substantial quantities of real world data.	**Yes**. SView can be used to browse any Linked Data on the Web. Heterogeneous data can be fused in a pay-as-you-go fashion.
3	The meaning of data has to play a central role.	**Yes**. SView gathers and deploys various views and navigators for conveniently browsing, understanding, and using Linked Data.

Table 2. Additional desirable features

	Feature	How we have addressed
1	The application provides an attractive and functional Web interface (for human users).	**Yes**. SView gathers and deploys both generic and domain-specific views and navigators for browsing and interacting with Linked Data.
2	The application should be scalable (in terms of the amount of data used and in terms of distributed components working together). Ideally, the application should use all data that is currently published on the Semantic Web.	**Yes**. SView can use all data that is published as Linked Data on the Web, and can fuse heterogeneous data from different sources in a pay-as-you-go fashion.
3	Rigorous evaluations have taken place that demonstrate the benefits of semantic technologies, or validate the results obtained.	**Yes**. We have reported a preliminary user study including a standard usability test.
4	Novelty, in applying semantic technology to a domain or task that have not been considered before.	**Yes**. SView is among the first to provide an open platform for browsing Linked Data, focusing on sharing views as well as pay-as-you-go data fusion.
5	Functionality is different from or goes beyond pure information retrieval.	**Yes**. For instance, SView supports browsing Linked Data via various views and navigators, and conducts pay-as-you-go data fusion.
6	The application has clear commercial potential and/or large existing user base.	**Yes**. For instance, developers can make a profit by embedding advertisements in their views. SView can easily find a way to make a profit based on its large user base in the future.
7	Contextual information is used for ratings or rankings.	**Yes**. For instance, views are triggered by the type or URI pattern of the present entity, or by the existence of particular properties in its RDF description.
8	Multimedia documents are used in some way.	**Yes**. For instance, an image view has been provided to display images about the present entity.
9	There is a use of dynamic data (e.g. workflows), perhaps in combination with static information.	**Yes**. For instance, our image view invokes the Flickr API to retrieve the latest photos on the fly.
10	The results should be as accurate as possible (e.g. use a ranking of results according to context).	**Yes**. For instance, entities can be filtered in a faceted manner.
11	There is support for multiple languages and accessibility on a range of devices.	**Yes**. Currently SView provides both English and Chinese versions.