Tabulator: Exploring and Analyzing linked data on the Semantic Web

Tim Berners-Lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets

Decentralized Information Group
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA, USA.

Abstract. A web of linked RDF data may be enabled by standards specifying how links should be made in RDF and under what conditions they should be followed as well as powerful generic RDF browsers that can traverse an open web of RDF resources. The Tabulator is an RDF browser, which is designed both for new users to provoke interest in the Semantic Web and give them a means to access and interact with the entire web of RDF data, and for developers of RDF content to provide incentive for them to post their data in RDF, to refine and promote RDF linking standards, and to let providers see how their data interacts with the rest of the Semantic Web. A challenge for Semantic Web browsers is to bring the power of domain-specific applications to a generic program when new unexpected domains can be encountered in real time. The Tabulator project is an attempt to demonstrate and utilize the power of linked RDF data with a user-friendly Semantic Web browser that is able to recognize and follow RDF links to other RDF resources based on the user's exploration and analysis.

1 Introduction

The Semantic Web[1] is a technology for sharing data, just as the hypertext Web is for sharing documents. The Web of Hypertext Markup Language (HTML) documents spread very quickly, partly because the author of an HTML document, or of a program producing HTML documents, could immediately see the results of his work. Further, by linking to other chosen resources, he could add immediate value to his page. This has not been true to the same extent with RDF data on the Web. While URIs are used for identifying things, the practice of supporting them with servers that would return relevant data, and links to other data, has been confined to use of URIs for properties and classes. Many projects such as Biopax [2], are currently available online but consist of a small number of huge archive files. There could be several reasons for this : (i) the emphasis during the development of OWL [3] on inference over fixed datasets, (ii) the chicken-and-egg problem with all network effects, that a lack of linked data is self-sustaining in that without things to link to, there is no incentive for

putting one's own data on the web, and (iii) the lack of a straightforward generic data browser, which would give an immediate feedback and gratification to the creator of online linked data.

There are many semantic web data browsers. Some of these, such as Palm-DAML[4], RDF Author[5] and IsaViz[6] are generic, but have a limited ability to display data in the form that a typical scientific or enterprise user expects, such as tables and graphs. Others such as CS-Aktive[7] and mSpace[8] provide a powerful browsing environment, but are tailored to a specific application area. These browsers are unable to automatically dereference web information by looking up URIs and thus do not support the browsing of linked data.

A goal of a generic semantic browser is **serendipitous re-use**: while searching for a solution to a problem, I discover and successfully use data, which I hadn't thought of using. I may not have known it existed, or I may not have realized that it was connected to the problem at hand. On the other hand, I may have known it existed, but not realized that it was now on the semantic web of linked data. A similar goal is that while idly browsing with a general interest in mind, I discover an interesting possibility which had not occurred before. The goal then is that, as with the HTML web, the value is the re-use of information in ways that are unforeseen by the publisher, and often unexpected by the consumer.

The Tabulator project [9] takes up this challenge of being a generic browser for linked data on the web without the expectation of providing as intuitive an interface as a domain-specific application such as calendar, money, or address book management. It hopes, however, to provide the sort of common user interface tools used in these kinds of applications, and to allow domain-specific functionality (such as views and forms) to be loaded transparently from the web, but most important, to be instantly applicable to any new domain of information. User interface metaphors are not obvious when navigation is possible simultaneously through web of links between data resources and the web of relationships of concepts. The project sheds light on, and raises more, questions in the domains of user interface design, the architecture of the web of data, and in the relationships between the two domains.

The project as an experiment is very much in the exploring mode. We did not carry out usability tests of the various browser forms. The motivations of the project include

- To make available to researchers and developers an in-browse software platform for new and derived RDF-based applications;
- To investigate the possibilities of, limitations to, and possible improvements in, Web architecture and Semantic web architecture;
- To investigate the use of existing and novel user interface metaphors in the context of the open data web;
- To find problems in general implementations or usage which need to be corrected or brought out in best practices documents;

– To find and design new breadcrumb protocols, that is, conventions by which pointers are left and followed in the Semantic Web so as to make certain classes of problems solvable.

2 Methodology

The intent of the project is to not only stimulate use of linked data but to further thought and development in the field of generic data interfaces as well. The browser should be as easy as possible for a new user to pick up, and easy for developers to extend with their own ideas. This requires as close to zero-install as possible, and development in a commonly understood and deployed language. The experimental platform was therefore developed as an in-page Javascript[10] application, and includes a Javascript library providing access to the web of data. The Asynchronous Javascript and RDF (AJAR) platform uses interfaces including the HTML DOM [11], HTTP, XML [12], RDF [13], and SPARQL [14].

It was considered essential to have the Tabulator run on the user's machine, so that that machine performs linked data protocols. While this raises difficulties, the intent was to explore and tackle those difficulties, so that protocols could be verified running across the net as opposed to being an HTML portal such as FOAF Explorer [15]. These difficulties include the need for user authentication to access (and merge) data of different sensitivities and the security and cross-browser scripting problem [16]. Though an HTML server side application is a tempting alternative design, this would not have been a decentralized design, not scaled with wide adoption, nor have allowed a growth in competitive and exchangeable data browsers.

2.1 Scenarios

We picked a number of scenarios as test cases to guide the development of the Tabulator project.

1. The documents published by W3C and its organizational structure are on separate web pages. As a working group chair, Alice wants a list of documents which are in Last Call status, their editors, and the email address of the domain leaders responsible for the working groups for which the document is a deliverable. She sorts them by date comments are due, and finds one she almost missed. She emails the chair of the working group asking for a extension of the deadline.
2. Bob has no idea what a line for $364 on his credit card statement is about. He looks at it on a calendar view, sees it was a Saturday, but still doesn't know. He looks for photos he took at the same time, and displays it on the same calendar. Now he understands: he took the kids to an amusement park.
3. Charlie notes that Danielle's blog is critical of his work, and wants to understand why that might be. Danielle has in her data file information pointing to papers she has written, projects she has worked on, and professional acquaintances. Charlie realizes they have an acquaintance in common who can probably help resolve the issue.

4. An existing relational database such as the W3C internal enterprise database, designed without semantic web export in mind, is exported into linked RDF. The Tabulator is found to be effective for browsing data in the database, both for the original users of the data, and also for new users.

3 User Interface

3.1 Web vs. Semantic Web

While a web browser navigates along links between documents, a semantic web browser navigates along relationships (predicates) in a web of concepts. However, a semantic web browser must also have an awareness of the underlying web of documents (and query services). Trust in information requires awareness of its provenance; access times depend on the number of remote accesses, and the user must be able to track and understand the effect of any network failures.

For the Tabulator the graph of logical information is primary, and the web of documents is secondary. The user explores an abstract web of data, which is the conjunction of all the graphs of documents that have been read. However, at any time he can check the provenance or source of any piece of information. Clicking on any data cell highlights the source in a pane of sources and their status. Double-clicking on the source allows the metadata about the source itself to be explored. Also, when information is associated with URIs which could be looked up, a small button is displayed whose color shows whether the data is not yet accessed (blue), already fetched (green), in progress (yellow), or failed (red).

3.2 Explore vs. Analyze

Another tension in user interface design is that between exploration and analysis. The WWW is something we explore, following links on a hunch that they will take us somewhere useful, reassessing the situation after each link. Exploring the semantic web in this way involves moving from node to node, finding more arcs, and reassessing our next move at each step. Most data applications, such as money management or calendar management, however, allow us to sort, process, and visualize data that has a very well-defined structure, such a set of bank transactions each with a date, payee, and amount.

To deal with these needs the Tabulator operates in two interlocking modes - exploration and analysis. The user can start using the Tabulator by submitting a URI, or by adding "?uri=URI" in the location bar of the browser. In the exploration mode, the user is unaware of what data is available, and can explore an RDF graph in a tree view, expanding nodes to get more information about them. As the user does this, the Tabulator implicitly follows links that may contain more RDF data about the relevant nodes. To move to the analysis mode, the user can select certain fields (arcs or predicates) to define a pattern, and ask the Tabulator to find all examples of that pattern. The Tabulator then performs

this query, following links as it attempts to match the query pattern to the RDF graph. The results of a query can be displayed in a number of modular views, including tables, and projections of time and space axes onto a calendar and map. These allow the dense presentation of data that one expects from domain-specific applications. One can start a new exploration from any table cell, or point in a calendar or map, and switch back to the exploration mode. If the subgraph pattern has temporal or spatial coordinates, these tables can be projected onto the calendar/timeline view or map view respectively. The user can double-click on any instance of a dense display of the selected cases to open a new exploration at that point. Exploration and analysis can therefore can be interleaved as the user tackles the task at hand.

3.3 Exploring in outliner mode

Many RDF visualizers such as IsaViz and RDF Author represent data as circle-and-arrow diagrams. This gives one a feel for how a small number of things connect. It allows one to see clustering when a large number of things are related by the same properties, such as in Foafnaut [17] or How/Why diagrams developed at W3C. Circles and arrows are very intuitive and useful when trying to understand the structure of data. However, it is not an appropriate way to look at data with many nodes and many different properties. It is not used in applications we think of as handling data, such as personal financial management, music management, calendar management programs, for example. In these cases tables or matrices are used. These are the densest way of comparing objects of the same class (strictly, which are likely to share common properties). mSpace is an example of a table-based semantic web browser. These table-based systems, though, tend to operate on a restricted set of data, and do not naturally allow browsing outside it.

The outliner mode of the Tabulator is an extremely common user interface metaphor and clearly natural for tree-oriented data. People are typically very comfortable in a tree-like environment. In fact, much data in the world has been organized into trees, and the web is largely composed of overlapping trees with local roots all over the place. This suggests that in fact a tree-oriented browser will feel natural even though the world is actually a web. In early hypertext, Peter Brown's Guide [18] system was an outliner, and the Gopher system [19] was a presented as tree (not in outline mode) though in fact it was a web. TreePlus, like outline mode, maps a graph to a tree, but it appears designed for graphs of homogeneous type, whereas Tabulator's outliner mode targets a heterogenous typed graph [20]. The outliner view uses a denser tabular format designed to meet user demands for more information to be visible at a time. Another related approach is Ghoniem's work [21], which deals with visualizing large complex graphs with single arc types but their matrix form is different from the table view used in Tabulator.

The outliner browser of the Tabulator is quite straightforward. Most of the design decisions involve how to represent options to load data, and whether to load it automatically - both of which have many variations. There are questions

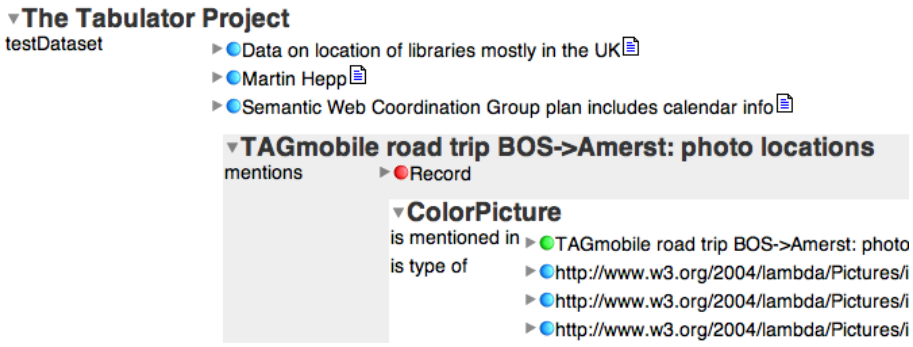

Fig. 1. Outliner View

as to whether to express links in both directions, and whether to suppress links which are reverse versions of a link displayed at a higher level. One meme of RDF ethos is that the direction one chooses for a given property is arbitrary: it doesn't matter whether one defines "parent" or "child", "employee" or "employer" [22]. In order to keep this independence on a specific direction, the browser must treat forward and backward links with equal stature. The outliner therefore displays links in both directions and is redundant, in that whenever an object is expanded more than one deep, for outer link will also be found as a "dual" inner link in the opposite direction. (e.g. Sam: father Joe: [is father of: Sam].)

3.4 Views of RDF data

While outliner mode offers a very general view of RDF data, Views can be used to emphasize specific properties of RDF data. The Tabulator provides several Views such as Map View, Table View, and Calendar View as different tabs. Underlying these views is a query-result management system. The results of the user's RDF graph queries are saved, and are made available for selection in any view. The results of different queries can be juxtaposed in the same Views to be compared. The Views are easily extensible; each view is an object that knows how to draw and undraw (remove) query information. The views also dynamically load the query results as they are found and returned.

Map View This view offers the user the advantage of displaying multiple queries on a single view in order to compare their geographical locations. The map view accomplishes this by implementing the Google Maps API [23] to plot RDF latitudes and longitudes that come out of a specific query. Additionally, the map view can handle addresses using the newly introduced geocoding feature in the Google Maps API. While some may be able to simply figure out a location by its address as given in a table view, very few users can make any sense of a highly accurate set of coordinates. With the map view, the user can overcome this barrier to get meaningful information out of simple coordinates.

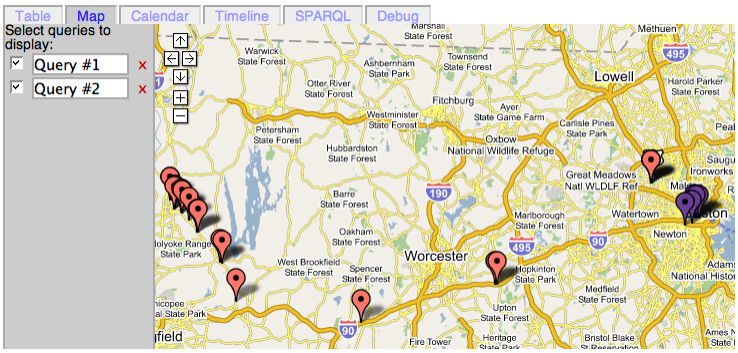

Fig. 2. Map View with selected restaurants

The following is an example that lends itself to the capabilities of the Tabulator's map view:

John wants to go out to dinner in Cambridge, but is not familiar with restaurants in the area. He does, however, know that he wants to go to a well-rated restaurant with meals under 20 dollars per plate. He also knows that he doesn't want to stray far from his hotel - the Marriott next to Kendall Square. John selects restaurants by their Zagat rating, their cost ($10-20), and their city (Cambridge) and tabulates them. He then plots his hotel as a single point on the map, and looks around for the restaurant he wants. The map view displays the restaurants as single points, and when John clicks on any of those points, he receives additional information such as the restaurant's Zagat rating, cost, and city in a bubble. Double clicking on the restaurant name opens its full description in the Outliner View.

Calendar and Timeline View The calendar view allows users to navigate between months with calendar data. Each day's events are sorted by time, and double clicking on the colored event cell will take the user back to the outliner view for more detailed and general information about the event. The calendar view is available when any the query results involve certain RDF predicates from the RDF iCalendar-equivalent ontology [24], which specifies vocabulary/ontology for working with other calendars.

The same RDF data that is viewable in calendar view is viewable in timeline view. The timeline view has been developed by Simile[25]. It has three colored bands (with scales of year, month, or day) that show the duration of the events at different scales. Clicking on timeline events opens up a bubble with additional information the user has highlighted. The user navigates the timeline view by dragging the bands; all three scales are geared together.

With the calendar and timeline views, the user is able to see when events overlap and visually assess event durations. For example, if a user wants to schedule a meeting for members of a group, the user can take the group's calendar

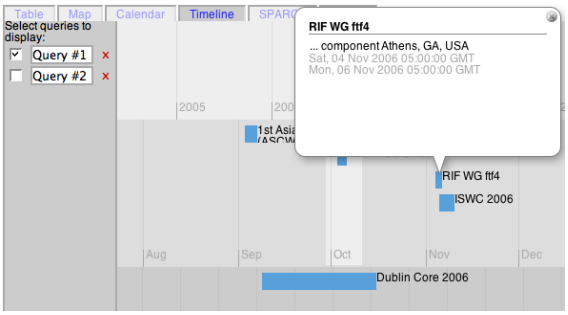

Fig. 3. Timelines shown at three different scales

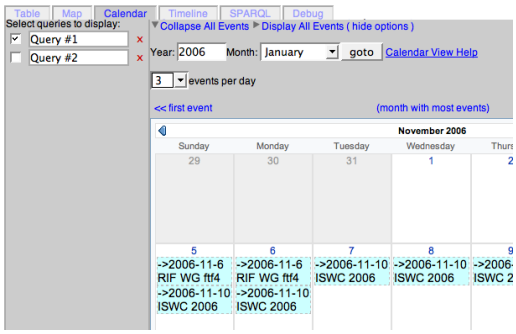

Fig. 4. Calendar View showing selected queries

events, plot them out on calendar view and timeline view, and look for gaps representing when the members are free.

SPARQL: Formulating Queries in Tabulator One of the big challenges in creating a simple user interface for Tabulator is allowing users to construct RDF query patterns. On the one hand, we would like the user to be able to construct a simple RDF graph query - for example querying the email addresses of all the Tabulator developers - without any experience or detailed knowledge of RDF. On the other hand, we would like a more experienced user to be able to construct more complex queries including variable constraints, optional triples, and cycles in the query pattern essentially, all queries allowed by the SPARQL specification.

One of the most user-friendly ways to construct queries is query-by-example. In the Tabulator's outliner view, an RDF graph is visualized in a tree structure. By highlighting a *property* field in this tree structure, the user specifies a query pattern based on the series of edges connecting this field to the root node. For example, browsing from the Tabulator project, opening a developer, and highlighting the schoolHomepage field will construct the pattern:

```
<#Tabulator> doap:developer ?v0.
?v0 foaf:schoolHomepage ?v1.
```

This can form the basis of an RDF query that will find all mailboxes of Tabulator developers. Simple constraints can also be added with query-by-example. Highlighting a value cell in the outliner view creates a query pattern as above, but also demands that the equivalent node in the RDF query have the same value as the highlighted cell. For example, highlighting someone's schoolHomepage would create the same pattern as above, with the constraint that

```
?v1 = <http://www.mit.edu>.
```

Finally, multiple fields can be highlighted to create more complex queries.

To allow expert users to define more complex queries, the Tabulator also allows the query to be viewed and edited in SPARQL itself.

4 Network access algorithms

The Tabulator looks up (dereferences) a URI whenever the user expands to find information about a subject in the outliner view. It also looks up URIs during the progress of a query. Every URI used in the query expression itself, and every URI that is bound to a variable during the query and therefore becomes part of the remaining query to be matched, is looked up.

The dereferencing of links in RDF has not been explored to a great extent to date, partly because of the fact that, if an inference engine simply downloads data whenever it comes across a new URI, it will, in an unbounded open web, continue without limit. This is the case with an undirected traversal of the semantic web. In the case of a user browser, however, there is direction, in the direction the user chooses to navigate, or in a specific query, which a user has posed.

The simplest linked data protocol is that for a given graph G, dereferencing the URI of any node x in G will return all arcs in and out of that node. This does not of course, help a server decide, for a given x what graph G would be useful to export. When a relational database is exported as RDF [26][27][28], the graph can be taken as the entire contents of the database. However, in the general case of an exploring user, the server has to anticipate what data will be useful. In fact, the limitations in practice turn out to be that some data would be just too numerous, such as giving, for a person, an entire track of every known date/location trackpoint. Another limitation is that not all data is actually available in machine-readable form. A general technique is to give either the data, or clues as to where to find it. A person's home page in HTML will not typically contain a full list of past employment, but can contain a link to one. This is similar to what can be done in RDF.

There are other protocols, and the Tabulator platform provides a platform for experimenting with different ones. We note however that an eventual goal would be to have common shared algorithms so that dependable results can be obtained. The project therefore kept closely to standards where they exist.

Though we currently focus on authoritative information, we hope to adapt Web procotols for non-authoritative information such as the use of annotation portals, and of query server descriptions.

4.1 From representation to RDF

The following outlines the algorithm used to extract RDF data from a representation of a resource. A representation fetched with HTTP has metadata, including a content-type, and a set of bits which may be parsed.

- The RDF/XML specification does not require an outer `<rdf:RDF>` element on a file, so when the content type is declared as application/rdf+xml, the representation can be parsed as RDF without rdf:RDF. However, if the content type is application/xml, then there is no authority to parse it as RDF unless its document element id `<rdf:RDF>`.
- If a representation is declared as text/html, then one can try to tidy it into an XHTML file, and then treat it as though it were.
- If a representation has content-type application/xhtml+xml, or is application/xml and has document element declaring it to be XHTML, then XHTML-specific operations are performed. Specifically, any `<link rel="meta">` link, if present, is dereferenced, and if the HTML profile attribute indicates that GRDDL (Gleaning Resource Descriptions from Dialects of Languages) [29] is being used, then the GRDDL specification is followed.
- If any XML representation has an attribute indicating that it follows the GRDDL specification, then GRDDL determines how a transformation resource (normally XSLT) is fetched and applied to the original representation.

These algorithms seem to represent the state of the art, though the GRDDL specification is in the process of standardization. Future work in the community to agree on this algorithm would be beneficial, although keeping it extensible through new internet media types is clearly important.

For future work, we would like investigate the use of a Link: header on an arbitrary HTTP resource, to provide the same functionality as the xHTML `<link rel="meta"/>` element in pointing to metadata about the resource. This would have the advantage that the client could perform a HEAD request, and could then choose just to access the metadata of, for example, a picture, without retrieving the whole image.

4.2 What to dereference

The Tabulator automatically and recursively loads the ontology file for any term used as a predicate or type (object of rdf:type), recursively (ontological closure).

Here we consider a user is browsing or querying information about a subject x. When the user opens up a tab asking for information on x, or a query is being resolved and x is the subject or object of a statement in the query pattern, or is x is bound to a variable during the query, then x is looked up. Looking up currently involves:

- looking up the URI of x itself, and also
- looking up any y where the store includes the fact that { x rdfs:seeAlso y}.

The latter is necessary for the Friend of A Friend (FOAF) conventions. It is currently not widely used elsewhere. It can, be useful, however, to allow a third party to point out that information is available, when the owner of the URI itself has not, for whatever reason, included that information when x is dereferenced.

We implemented the dereferencing of URIs using the HTTP protocol. Successful dereferencing of an HTTP URI gives a status code and either a redirection (status 300-303) or (status 200) a representation consisting of content bytes and metadata.

4.3 Hash signs and redirects

When a URI containing a hash (#) is dereferenced, the URI looked up is the part to the left of the #. This allows information on many things to be defined in the same file. Some systems use URIs without a #. In this case, if the URI denotes an arbitrary thing, then the correct server behavior is to respond with a HTTP 303 response, forwarding the client to a document which does contain information. The Tabulator will follow this protocol [30].

A snag at this point for URIs that do not have hashes is that the Tabulator launches a fetch for every URI. For the Dublin Core [31] namespace, for example, it will launch fetches for each of the properties dc:title, dc:author, etc, and receive for each a redirection to the same place. The current (2006/7) Tabulator code includes special case overrides for the Dublin Core and FOAF namespaces for this reason.

4.4 Friend-of-a-friend conventions

The FOAF[32] world is an existing and growing world of people who have published small RDF files about themselves and their work (and sometimes other aspects of life) . The FOAF convention is to identify a person indirectly by their email address or, for privacy, the checksum of their email address. Also given, by rdfs:seeAlso, is a link to their FOAF file. The protocol, then, is then to load the resource linked by rdfs:seeAlso, and then to merge ('smush') nodes with the same mailbox or mailbox hash. As these are inverse-functional properties, they are dealt with by the inference layer, which is described next.

4.5 Inference on the client

Another issue involves deciding the kinds of inference that need to be performed. This is part of the network protocol, in that the duty of the server to provide "useful" data in response to a query about something is met more easily (and with less bandwidth) if the server can assume that the client is performing the inference. However, adding inference to the client slows it down, and speed in a browsing user interface is a very important factor.

The limited inference performed is as follows.

- owl:sameAs smushing: merging nodes which are owl:sameAs each other. Merging of equal nodes is often the only way to merge data from different sources. owl:sameAs is rarely used directly.
- Merging nodes with identical Functional or Inverse functional properties. Essential for the FOAF convention, and very useful for others.
- rdfs:subPropertyOf (currently only) used for finding subproperties of rdfs:label for the user interface.
- URI canonicalization Certain URIs have only syntactic differences and are always equivalent. URIs are canonicalized by the store.
- Hashed property entailment: { `?p :Sha1Property ?q. ?x ?p y?. ?y crypto:sha1 ?z}` `=>` {`?x ?q ?z`} This is needed for matching foaf:mailbox to foaf:sha1_mbox. (Not currently implemented 2006/7)

OWL-DL entailment is not supported.

5 Related Work

The Tabulator builds on a long tradition on data presentation programs.

Popular among them are applications which are domain-specific semantic data browsers. Such applications are basically a front end tailored specifically for a particular RDF database. mSpace has a database of classical music detailing classical music pieces, composers. Users can sort by era, time, composer, type of composition etc and then listen to there selections. It doesn't have links to any outside database but it does have a window that will display google.com search results relevant to the user's music selections.

[33] is the website for the all the art museums in Finland collectively. It has all the museums' inventories cataloged in RDF and a website through which users can link to similar inventory items based on parameters such as time, artist, country, and other typical fields.

These applications showcase a powerful way to organize data using RDF and they have incredibly user-friendly front ends. Eventually, Tabulator should be able to provide a generic front end for these datasets.

There also exist applications that are a front end for a particular RDF dataset but allow the user to both browse and analyze the data. CS AKtive[7] is has a database of people with information about their field of research and projects. Not only can the user see this data in organized lists, but can, for instance,

select areas on a map and return people within a given radius. Another tool for browsing and analyzing a database of people is Flink. They have analytical tools that map out the individuals in its database on several different types of maps and allow for navigation of connected individuals in a web-like structure. Personal Publication Reader is an application which has browsing and analytical tools on a database not of people but of research papers, primarily.

Some applications including Magpie [34], Haystack [35], Piggybank [36], and Longwell [37] do generic semantic data browsing. All four are applications that a user must download on their machine. Piggybank relies on screen scrapers to collect data that it can interpret and analyze (for example, collecting addresses for a website, understanding that they can be put on a map and then mapping them. Magpie is similar except it will search a webpage for word or other elements which it think it can create meaningful links from, it is also not dependent on RDF data. Haystack is a generic RDF-based system which gives a powerful set of views of online data. It does not explicitly follow linked data, being primarily an experiment in generic application building using RDF techniques. Longwell is a faceted browser that allows users to filter RDF data by creating constraints on the data. Though this is similar to Tabulator's query interface, Longwell runs over pre-loaded RDF data and does not perform any linked data protocols.

6 Future Work

The Tabulator project has led to many more questions than it has answered. Medium term extensions include Notation3 [38] parsing, various RDF (and full N3) document content views, the implementation of the Fresnel [39] lens and style language. Our future work may include extension of query-by-example to intuitive rule-building systems, and experiments with a variety of remote stores and query engines. We may also provide a more graphical user interface using Scalable Vector Graphics (SVG) or possibly even 3D.

Given the goal of the web as an interactive space for common creativity, an essential development is for the user to be able to intuitively modify and extend the data during a browsing session. Extensions of the Fresnel language to cover input form views are one possibility, while another is to guess what properties a user may want to add from the existing data. A design decision then is whether to make the client save back a modified file, as conventional web editors do, or, more interestingly, to share every change over the network with a server, and other clients viewing or editing the same information.

7 Evaluation

Though we did not carry out extensive usability testing, we did encourage W3C staff members to use and evaluate Tabulator. Their feedback helped us understand the problems and requirements of technical users. Several of their comments were incorporated into the new release of Tabulator such as their request for a dense representation of information. The most common comments include

: (i) no way to undo or go back to a certain previous state, (ii) lack of cognitive map, and (iii) requirement of spreadsheet like navigation.

8 Conclusions

The architecture of linked data proves to be a powerful one, and it is possible to build a generic data browser that provides sufficient functionality to make new data on the Semantic Web immediately viewable. The technique of providing general views of arbitrary data that interlock seamlessly with specific views for particular special cases such as time and space seems to be a sweet spot between a completely generic interface and a completely application-specific one. That said, domain-specific applications will always be important and will always do better at specific tasks than the general one. This suggests that smooth interoperability between a generic client and an application-specific one is crucial.

Perhaps the past lack of development of linked data is due to the fact that a harvester following links in general will attempt to load an unbounded set of data. However, in both tabulator modes, exploration and analysis, this is not the case, as information is loaded to met the curiosity of the user. The technique rests on the use of existing, impending, and putative standards to determine the algorithms for serving and retrieving data. These standards will mature with time, but the experience presented here indicates that the overall algorithm, and the GRDDL specification that forms part of it, would benefit from documented consensus.

We believe that in order to enable generic browsing, users should be encouraged to leave sufficiently powerful user interface tips in ontologies so that a generic application can acquire, in real time, the ability to provide an effective and useful interface to data from previously unknown domains.

References

1. Herman, I.: The Semantic Web home page. `http://www.w3.org/2001/sw/` (2006)
2. Sander, C.: BioPAX Level 2, Version 1.0. `http://www.biopax.org/` (2005)
3. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language Reference. Technical Report http://www.w3.org/TR/2004/REC-owl-ref-20040210/, W3C (2004)
4. Dean, M., Margerison, J.: PalmDAML. `http://www.daml.org/PalmDAML/` (2002)
5. Steer, D.: RDFAuthor. `http://rdfweb.org/people/damian/RDFAuthor/` (2003)
6. Pietriga, E.: IsaViz. `http://www.w3.org/2001/11/IsaViz/` (2006)
7. Glaser, H., Alani, H., Carr, L., Chapman, S., Ciravegna, F., Dingli, A., Gibbins, N., Harris, S., m. c. schraefel, Shadbolt, N. In: CS AKTive Space: Building a Semantic Web Application. Springer Verlag (2004) 417–432
8. m. c. schraefel, Karam, M., Zhao, S.: mSpace: interaction design for user-determined, adaptable domain exploration in hypermedia. In: AH 2003: Workshop on Adaptive Hypermedia and Adaptive Web Based Systems, Nottingham, UK (2003) 217–235

9. Berners-Lee, T.: Tabulator Project. http://dig.csail.mit.edu/2005/ajar/release/tabulator/0.7/tab.html (2006)
10. ECMA: ECMAScript Language Specification. Technical Report ECMA-262 (1999)
11. Stenback, J., Hëgaret, P.L., Hors, A.L.: Document Object Model (DOM) Level 2 HTML Specification, Version 1.0, W3C Recommendation 09 January 2003. http://www.w3.org/TR/DOM-Level-2-HTML/ (2003)
12. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F.: Extensible Markup Language (XML), 1.0 (Third Edition), W3C Recommendation 04 February 2004. http://www.w3.org/TR/2004/REC-xml-20040204/ (2004)
13. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-concepts/ (2004)
14. Seaborne, A., Prud'hommeaux, E.: SPARQL Query Language for RDF. Technical Report http://www.w3.org/TR/2006/CR-rdf-sparql-query-20060406/, W3C (2006)
15. Frederiksen, M., Dodds, L.: FoaF Explorer. http://xml.mfd-consult.dk/foaf/explorer/ (2002)
16. Wikipedia: Same origin policy. http://en.wikipedia.org/wiki/Same_origin_policy (2006)
17. Ley, J.: FOAFNaut. http://www.foafnaut.org/ (2006)
18. Brown, P.J.: Turning Ideas into Products: The Guide System. In: Hypertext, ACM (1987) 33–40
19. McCahill, M.: The Internet Gopher: A Distributed Server Information System. ConneXions - The Interoperability Report **6**(7) (1992) 10–14
20. Lee, B., Parr, C., Plaisant, C., Bederson, B., Veksler, V., Gray, W., Kotfila, C.: TreePlus: Interactive Exploration of Networks with Enhanced Tree Layouts. IEEE TVCG Special Issue on Visual Analytics (2006)
21. Ghoniem, M., Fekete, J.D., Castagliola, P.: A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations. In: INFOVIS '04: Proceedings of the IEEE Symposium on Information Visualization. (2004)
22. Berners-Lee, T.: Enquire V 1.1 Manual. http://www.w3.org/History/1980/Enquire/ (1980)
23. Team, G.M.: Google Maps API. http://www.google.com/apis/maps/ (2006)
24. W3C: iCal Schema. http://www.w3.org/2002/12/cal/icaltzd (2005)
25. Huynh, D.F.: SIMILE — Timeline. http://simile.mit.edu/timeline/ (2006)
26. Berners-Lee, T.: Relational Databases on the Semantic Web. http://www.w3.org/DesignIssues/RDB-RDF.html (1998)
27. Connolly, D.: Exporting databases in the Semantic Web with SPARQL, D2R, dbview, ARC, and such. http://dig.csail.mit.edu/breadcrumbs/node/140 (2006)
28. Connolly, D., Crowell, R.: dbview.py – HTTP access to an SQL DB thru RDF glasses. http://dig.csail.mit.edu/2006/dbview/dbview.py (2006)
29. Hazaël-Massieux, D., Connolly, D.: Gleaning Resource Descriptions from Dialects of Languages (GRDDL), W3C Team Submission 16 May 2005. http://www.w3.org/TeamSubmission/grddl/ (2005)
30. Fielding, R.T.: [httpRange-14] Resolved. http://www.w3.org/mid/3fc8037bc096da8c801ebc8c1295e09b@gbiv.com (2005)
31. Beckett, D., Miller, E., Brickley, D.: Expressing Simple Dublin Core in RDF/XML. Technical Report http://dublincore.org/documents/2002/07/31/dcmes-xml/, Dublin Core Metadata Initiative (2002)
32. Brickley, D., Miller, L.: Friend of a Friend (FOAF) project. http://www.foaf-project.org/ (2000)

33. Hyvönen, E., Junnila, M., Kettula, S., Mäkelä, E., Saarela, S., Salminen, M., Syreeni, A., Valo, A., Viljanen, K.: Finnish Museums on the Semantic Web. User's Perspective on Museum Finland. In: Proceedings of Museums and the Web 2004 (MW2004). (2004)
34. Dzbor, M., Domingue, J., Motta, E.: Magpie: Towards a Semantic Web Browser. In: Proc. of the 2nd Intl. Semantic Web Conf. (ISWC). (2003)
35. Quan, D., Huynh, D., Karger, D.R.: Haystack: A Platform for Authoring End User Semantic Web Applications. In: ISWC. (2003)
36. Huynh, D., Mazzocchi, S., Karger, D.: Piggy Bank: Experience the Semantic Web Inside Your Web Browser. In: International Semantic Web Conference (ISWC). (2005)
37. Butler, M., Huynh, D., Hyde, B., Lee, R., Mazzocchi, S.: Longwell project page. `http://simile.mit.edu/wiki2/Longwell` (2006)
38. Berners-Lee, T.: Notation 3 (N3) : A Readable RDF Syntax. `http://www.w3.org/DesignIssues/Notation3.html` (1998)
39. Bizer, C., Garland, S., Huynh, D., Karger, D., Lee, R., Mazzocchi, S., Pietriga, E., Quan, D., Bakshi, K.: Fresnel - Display Vocabulary for RDF. `http://www.w3.org/2005/04/fresnel-info/` (2006)