Interactively using Semantic Web knowledge: Creating scalable abstractions with FacetOntology

Daniel Alexander Smith
IAM Group
School of Electronics and Computer Science
University of Southampton
Southampton, United Kingdom
das05r@ecs.soton.ac.uk

mc schraefel
IAM Group
School of Electronics and Computer Science
University of Southampton
Southampton, United Kingdom
mc@ecs.soton.ac.uk

ABSTRACT

The amount of knowledge accessible on the Semantic Web is growing, and there is a need for a scalable solution to facilitate exploring that data. Currently approaches to exploring Semantic Web data either focus on exploring resources individually, following links during exploration, and making little use of collated data, or take the approach of collating and aligning multiple sources into one store for one purpose, and hand-crafting a specific browsing interface onto it. We present an approach that provides a scalable browsing interface, which can browse knowledge from the Semantic Web at will. Our approach creates abstractions of knowledge, collated into facets, which are described using FacetOntology. FacetOntology facilitates descibing facets from RDF data, suitable for use in creating datasets for faceted browsing.

Categories and Subject Descriptors

H.6.m [**HCI**]: Miscellaneous; D.2.2 [**Software Engineering**]: Design Tools and Techniques

Keywords

Semantic Web, browsing, scalability, abstraction, alignment, knowledge

1. INTRODUCTION

There is a growing amount of information being published to the Semantic Web, through services such as DBPedia[1] and Freebase[1] that provide Semantic Web APIs to their large datasets, and by the open linked data community[2] that are generating links between such data sets. As this amount of interlinked data grows, being able to explore and browse it is becoming more of a challenge. When the Semantic Web was in its infancy, browsers that allowed manual walking between personal data like Friend of a Friend**??** provided a tractable user experience, whereas scaling this up to exploring all of DBPedia, Freebase or open linked data doesn't work: there is now so much information that manually exploring resource-by-resource is inefficient, and does not make full use of the data available.

[1]Freebase: `http://rdf.freebase.com/`
[2]Linked Data: `http://linkeddata.org/`

Research into development of browsers to explore data from the Semantic Web has resulted in two different approaches for browsing data on or from the Semantic Web. The first approach is to directly browse the linked data structure of the web, whereby a user inputs a URI of a resource as a starting point, and the interface exposes links in that document to other resources on the Semantic Web, allowing the user to browse those. Browsers such as the Disco Hyperdata Browser[3] and the Tabulator[2] take this approach, which benefits from browsing the Semantic Web directly, much like browsing the World Wide Web, in that no centralised server is required. The second approach of browsers is to rely on pre-processed, harvested data, either from the Semantic Web, or taken from other other sources and manipulated into Semantic Web knowledge. This knowledge is then loaded into a storage and query system particular to the browser, then optimised and indexed for an interactive level of speed. This approach benefits in speed and stability, in that all of the data is loaded and indexed ahead of time, and as such, query time is optimal, and the browser is not affected by third-party sites going down.

A model that we propose is a mixed-source browsing interface, whereby an exploratory browser can support querying both pre-processed data, and raw RDF or SPARQL data on the Semantic Web, and adding found data to a processing queue, for faster indexed access in the future. Such a system would combine the benefits of using data that has been optimised for browsing, with the ability to explore live data on the Semantic Web.

One of the expectations placed on such a system is that the data used is up-to-date, as users would be used to this in the live-browsing variant of the browser. As such, the infrastructure to support the browser has to employ a level of complexity on the server side, particularly with the updating of data, much like a search engine on the web has to do.

Once data has been collated, the faceted browser ontology must be applied, in order to create specific abstractions over the combined set of data to be used by faceted browsers.

In Section 2 we discuss the problems with browsing data from the Semantic Web: aligning multiple data sources, speed and scalability, and creating an abstraction over RDF that is suitable for browsing. In Section 3 we discuss the tools and technologies that allow data to be exposed to the Semantic Web, to be transformed and to be accessed. In Section 4 we discuss three existing Semantic Web browsing

[3]Disco `http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/`

tools, and their techniques for facilitating data access and browsing. In Section 5 we discuss our approach to browsing the Semantic Web, and how it utilises a multiple-data-sources strategy to enable scalable browsing of data from the Semantic Web. Section 6 discusses future work on implementation, and Section 7 summarises our conclusions.

2. EXPLORING THE SEMANTIC WEB

In this section we discuss three main problems associated with browsing and exploring data from the Semantic Web, work that is related to each problem, and how our approach addresses that problem. In Section 2.1 we discuss the problems associated with multi-source alignment, in Section 2.2 we discuss the problems of speed and scalability of browsers, and in Section 2.3 we discuss how to define an abstraction over data that is suitable for browsing.

2.1 Multi-source alignment

In this section we discuss the problems that are encountered when aligning data from multiple heterogeneous sources. In Section 2.1.1 we discuss the problems associated with ontology alignment, and in Section 2.1.2 we discuss the problems associated with co-reference. We also discuss the related work, and how our approach deals with the problems.

2.1.1 Ontology alignment

Ontology alignment is a longstanding research challenge [6, 20, 22] that affects multiple fields of computer science, library science and linguistics, to name but a few. The problem of ontology alignment is that ontologies that represent the same domain may vary in structure and symbology, as equivalent terms are represented with alternative URIs. This variation depends on two main factors; the intended application of the data, and the granularity to which an ontology represents its domain.

In a typical ontology alignment example, three data providers that publish temperatures from physical weather stations, where each use their own ontology to describe the temperature data. A harvester would be unable to collate and present a set of unified temperatures from these data providers, because the data is described differently. In particular, the data providers' ontologies have a different class hierarchy whereby the temperature unit is defined at different depths, described using a different predicate URI, and in two different units: Celsius and Fahrenheit. In combining these sources, the two providers that publish temperatures in Celsius will need to have a mapping between their ontologies defined, so that data harvesters are aware that the data has the same semantics and should be represented in the same way on the browsing interface. The data from the source that published in Fahrenheit will have to also require a mapping to be defined, and additionally a rule that defines the mathematical relationship between the Fahrenheit and Celsius units of measurement, so that the data can be correctly converted into a common unit of measurement, on the interface of a browser. The ontology alignment challenge is the ability to correctly align semantically equivalent concepts. This challenge varies for different comparative ontologies, and can become complex whether the process is manual, semi-automated or fully automated. For example, two ontologies may represent a semantically equivalent class hierarchy, however one may express a greater level of granularity, by providing a deeper class hierarchy. The ontology alignment process then aims to map the classes together, and preferably into an abstraction that does not break the conceptual model of either ontology, if possible. Ontology alignment is a relatively new field, and the accuracy of semi-automated and automated alignment techniques can be unreliable [15]. Using alignment techniques can provide erroneous results, and hence can prevent faceted browsers from identifying relevant data sources and incorrectly identify false positives, which lead to data sets with incorrect representation of a domain.

In particular, the Alignment API[6], utilises a number of ontology matching techniques, and exposes them via a software framework that enables finding, storing and applying alignments across different data sources. Other ontology alignment frameworks also exist, such as PROMPT[20], which also finds conflicts and makes suggestions about possible alignments.

This research could provide faceted browsers with exploitable data sources. To date this gap in the field has not been explored: no browser has explicitly made *automated* ontology alignment a focus of the browser. CS AKTive Space[17] utilised data from multiple sources, and employed the technique of mapping each data source to a single ontology, which was used to drive the exploration of the data. Potluck[12] provided users with an easy-to-use data mapping tool, that allowed so-called "semi-ontology alignment for casual users." The interface enabled users to specify sources of data from the web, and manually describe which attributes from the data described the same thing (for example, that one web site's data on "phone number" was the same as another web site's data on "telephone"). Our approach supposes that exploring heterogeneous sources over the Semantic Web exposes the user to ontology alignment problems frequently, and as such the inclusion of an automated or semi-automated alignment feature would be beneficial.

2.1.2 Co-referential integrity

Ensuring co-referetial integrity is also a well known problem, which differs from ontology alignment in that it refers to determining if two instances are equivalent, as opposed to two classes or predicates. For example, two data sources may provide metadata about the same city: London, the capital of the UK. However, because they use different URIs to refer to the city, there is a co-referential integrity problem, and a harvester is not aware their resources refer to the same instance. This erroneously leads to two different instances being represented in a browser, rather than combining the information from both sources into a single instance that represents London. The problem of co-reference can encompass determining an equivalence mapping between two resources that describe the same instance, to more complex issues of granularity differences, for example between two resources that describe different roles of a person, where those roles may have contradictory attributes, and therefore combining the attributes from those roles into a single instance would be invalid. Co-reference is therefore a problem for semantic web explorers, as it is important that two resources that refer to the same instance are marked as equivalent, so that related facets are collated, and the exploration is accurate.

There are a number of co-reference solutions that vary from looking at domain-specific solutions in metadata, to generic solutions that aim to work for any data. Domain-specific solutions work by knowing facts and patterns in metadata that apply to particular domains. A pertinent example is AKTiveAuthor [19] which takes citation metadata from repositories, and attempts to provide semantic equivalance between authors of different papers. AKTiveAuthor supposes the hypothesis that authors often cite themselves, and hence that when a citation between papers is made, and there is a lexical match between an author of the paper and an author of the paper cited, those authors are the same person, and a semantic equivalence should be made. This kind of contextual knowledge can be applied to different domains, but if there is no knowledge available, a generic solution will be necessary [7]. In terms of facilitating the look-up and disambiguation of co-reference of resources, the concept of a Co-Reference Service (CRS) is discussed and a prototype is presented in [14]. This work states that in order to successfully manage co-reference on a large scale, such a service should not return definite equivalences, and that it should instead indicate the membership of an instance in a "co-reference bundle", which indicates a high probability of equivalence.

By ensuring co-referential integrity of instances, semantic web explorers can ensure that they do not erroneously create multiple references on an interface to the same resource. This problem is similar to that of Ontology Alignment, as discussed in Section 2.1.1, in that no exploratory browsers that utilise Semantic Web data bring the issue of co-reference to the forefront within the browser. The Tabulator[2] supports utilising explicit equivalence (in particular, equivalence defined by `owl:sameAs`, and `[inverse] FunctionalProperty`)[4], however no techniques for determining unspecified co-reference are used. Our approach suggests that an exploratory browser of semantic web data would benefit from utilising techniques to determine co-reference of resources, either internally or via third-party co-reference services.

2.2 Speed and scalability of browsing

One of the problems in browsing distributed data sources is getting the data to the client in a sufficiently fast enough time to provide an adequate user experience. Browsers that collect and process data from the Semantic Web, at the point of exploration, do not have the capability to provide an adequate response time once the scale of data surpasses that of personal metadata.

We describe scalability in terms of a browser delivering the data user requires without overloading the following:

- Server-side
- Client-side
- User cognition

In order to provide a scalable solution to this problem, browsers have looked to the use of queryable knowledge bases as a way to deliver low-response-time queries to exploratory browsers. For example, both CS AKTive Space[17] and /facet[10] utilised this approach, whereby a large volume of data was asserted into a triplestore (using 3store[8] and SWI-Prolog[28], respectively), and a query language was used by the browser to live query the triplestore for results matching the constraints selected by the user in the interface. While these solutions enable medium-scale data to presented to browsers, once large-scale data access is required, a return to traditional more scalable relational databases is required, as shown by mSpace, which compared different storage and query systems for interactive speed over large data sets[24]. While these solutions enable large volumes of data to be queried, none of them allow on-the-fly access to the Semantic Web.

By enabling access to additional Semantic Web documents, browsers can increase the amount of data that they display, and access additional instances and domains of data on-the-fly. Our approach suggests the model of a mixed-source browsing interface, whereby an exploratory browser can support querying both pre-processed data, and raw RDF or SPARQL data on the Semantic Web, and adding found data to a processing queue, for faster indexed access in the future. Such a system would combine the benefits of using data that has been optimised for browsing, with the ability to explore live data on the Semantic Web.

2.3 Creating an abstraction suitable for browsing

Knowledge harvested from the Semantic Web is rich for exploration. The level of sophistication by which this knowledge is presented for exploration can vary. Basic browsers such as Disco[5] allow direct browsing of a resource, given its URI, and browsing to other resources, by the links present in the RDF of the resource. This paradigm is taken further by Tabulator[2], which additionally allows a user to graphically define a pattern in the data, and tabulate the results of that query, when run over all loaded RDF documents. The tabulated results can then be viewed in different ways, depending on the content of the metadata, for example, latitude and longitude data can be viewed on a map, and dates can be viewed on a calendar.

The problem is that while browsing the semantic web directly respects the distributed nature of the web, and highlights provenance of documents and utilises the structure of the semantic web in a tangible way, the user experience can be improved. The improvements come in the form of collating multiple documents and allowing the user to browse an abstraction of the combined data that fulfils their needs.

There is a trade-off that exists on this spectrum of knowledge browsers, in how much description of the schemas of the data must be performed, versus how user-friendly and understandable the browsing experience will be. As noted above, the Disco hyperdata browser provides browsing of RDF sources, with no configuration necessary. The user simply enters the URI that they wish to explore, and this is dereferenced, an RDF document retrieved, and the knowledge directly linked to the URI is then rendered and exposed. In order to make the graph walking easier, the labels of connected instances are resolved and displayed, a process that takes place in the background and updates continuously. The browser also provides google maps boxes for geographic co-ordinates and downloads images that are referenced, for example in *foaf:depicts* triples.

No additional sematically-specific rendering is performed, such as was explored in the Fresnel project [21]. *Fresnel* defines the concepts of *lenses*, which are human created de-

[4]Tabulator presentation: `http://dig.csail.mit.edu/2007/Talks/0511-tab-tbl/`

[5]Disco `http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/`

scriptions of data, that specify for a given collection of properties, render them to XHTML in a certain fashion. For example, a simple lens might define how to create a piece of XHTML that makes data marked up in the FOAF[4] or vCard[5] ontologies look like a traditional "business card." This area of research deals with how to render specific instance data, as opposed to creating an abstraction over the ontological structure of RDF data, but it is potentially useful as an example. This is because it takes the stance that "given that property X is used, query for property Y from a connected instance and render them together," which is a familiar argument, as it is similar to that which is used when describing data for *mSpace*, or when abstracting data out for simplified viewing in */facet*.

Creating an abstraction over data for */facet* is an important topic to note. */facet*'s main contribution is the creation over a faceted interface over semantic data without the requirement of a configuration to describe the schema of the data, and how the ontology maps to facets. Unlike the *mSpace* approach, where a single goal type is chosen and with all facets described in terms of their graph pattern match to that goal type. In */facet*, the user chooses a type at the start of the interaction, an approach used, for example in OntoBrowse [23] software from 2004, where the user selects a property from a list of all properties in a knowledge base, and browses all instances that hang off that property. Once the user has selected a type, all properties that connect from that type are shown and collated as separate facet columns.

While the method */facet* employs does create a faceted browser quickly, it exposes the inner normalised ontological hierarchy to the user, rather than more usable compressed lists, as one would typically find in faceted browsers, as hierarchies with too deep layers are confusing to users without domain knowledge[9]. There is work into the creation of automatic shallow hierarchies for faceted browsing, such as Castanet[25], which has yet to be applied in the general case, with Wordnet being shown as an example corpus. Currently, in order to provide a cognitively acceptable interface using */facet*, it would be necessary to flatten deep hierarchies and combine properties so that instances that one would want in facets are a single property arc away from each RDF type that the user can choose. This could be performed a number of ways, such as using reasoning in OWL[11], possibly combined with RIF[27] rules where necessary. A simple of example of where RIF is useful over OWL is when converting mathematically between scientific units, such as from Fahrenheit to Celcius. This kind of manual description of the data would be almost identical to the kind of description that *mSpace* requires in its *mSpace Model* definition format, and as such, */facet* is in fact not less work than mSpace to get going well than one might have thought.

This issue highlights the core challenge of this research area, of how to create appropriate abstractions over rich and combined data sources, that are suitable for exploration. These abstractions may be dynamic or fixed, and they may be pre-computed or dynamically created on the fly.

Our approach suggests an ontology that enables the attributes that define facets of explorable data to be defined, such that a faceted browser can present these facets in an interface, as an explorable abstraction of the data source.

3. SEMANTIC WEB DATA TRANSFORMATION TOOLS AND TECHNIQUES

Research into the Semantic Web has yielded a number of methods of exposing data to the Semantic Web, such as creating linked data, or exposing a SPARQL end-point, each with their own intended uses and benefits. Publishing data using resolvable URIs that return RDF (hereafter referred to as Linked Data), allows data to be linked across the web, and crawled by software agents and by humans using exploratory interfaces. Exposing a query end-point using SPARQL allows large volumes of triples to be queried, and results returned in a tabular format, which can be more appropriate to some data sources than exposing only linked data.

Tools are available to transform data from one form to another. For example Pubby[6] enables linked data to be exposed from a SPARQL end-point and D2R Server[3] creates a SPARQL query end-point to data that is stored in a relational database. These tools can act as a bridge between data access methods, enabling an end-user interface that supports only one data access method (such as supporting querying through SPARQL, but not the reading of RDF files), to access data exposed via a different method. The price of doing so is the time is takes to get the data to the client; the more live transformation that data requires, the longer it will take for the client to receive the data, and the worse the user experience will be. We have collated and presented the most commonly used tools that transform data between interfaces, and graded their interactive performance, based on our experiences in using the tools to back-end a semantic web project.

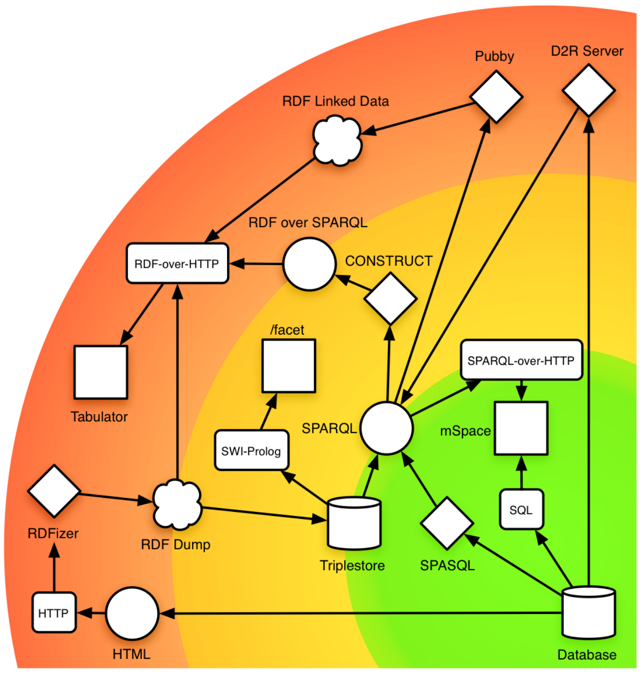


Figure 1: Interactive performance of Semantic Web data access and knowledge transformation technologies.

[6]Pubby: `http://www4.wiwiss.fu-berlin.de/pubby/`

In Figure 1 we present an overview of commonly used Semantic Web tools and technologies available to transform and access knowledge. Standard ways of exposing knowledge on the semantic web (namely RDF-over-HTTP and SPARQL) are plotted as symbols, as are commonly used tools to transform data from different sources to RDF and SPARQL. Three interfaces (mSpace, /facet and Tabulator) are also plotted, showing which data access methods they support. The symbols are plotted onto concentric coloured circles of red, yellow and green, which indicate the realtime performance of the represented system from slow, to medium and to fast, respectively. Several systems are plotted on the borders between colour zones as their performance is non-linear, and can lie in either zone, depending on use.

The figure details different data presentation protocols as rounded rectangles (SQL, SPARQL-over-HTTP, SWI-Prolog and RDF-over-HTTP, HTTP), data transformation systems as diamonds (SPASQL, RDFizer, CONSTRUCT, D2R Sever and Pubby), user interfaces as squares (mSpace, /facet and Tabulator), static RDF as clouds (RDF Linked Data, RDF Dump), live query interfaces as circles (RDF over SPARQL, SPARQL, HTML) and storage systems as cylinders (Database and Triplestore). Possible methods of transformation of data from one form to another are denoted by directional arrows.

3.1 Data access compatibility

While the technologies shown in Figure 1 illustrate that compatibility between different data access methods is possible, it also shows that many of the tools to transform data perform very poorly in terms of interactive speed, fundamentally preventing them from being usable in a live query environment behind an interactive interface. This is not surprising, as there is nothing about the Semantic Web that promises speed of query response over distributed sources, and this remains a well established research challenge. Consequently, in order to provide a sophisticated browsing experience, data has to be harvested and pre-processed ahead of time, allowing sufficient matching to be performed over heterogeneous data, and of interface-specific indexes of the data to be produced.

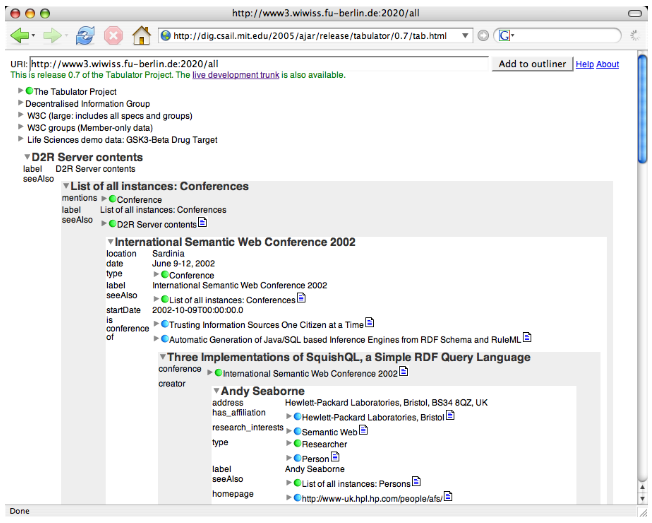
4. EXAMPLES OF SUPPORT OF EXISTING BROWSING TOOLS

Research into the Semantic Web has yielded numerous tools for browsing, editing, inference and publication. Where there are ambiguities in the support of certain criteria (as outlined above), solutions have been implemented in a decentralised and ad-hoc basis. The differences and subtleties in the tools are outlined in this section.

4.1 Tabulator

Tabulator is the canonical example of a Semantic Web browsing interface. It takes a URI as the starting point of browsing, and exposes the resources described by the RDF it downloads as a tree (as a 2D representation of a graph). Any resources that are identified by a URI can be followed and their URIs will be resolved, and the resources their RDF describes are then put into the tree at the appropriate place in a recursive manner, as the user clicks, as shown in Figure 2.

Tabulator's functionality provides abstraction over the browsed RDF at the time of browsing. The user must walk the RDF


```
http://www3.wiwiss.fu-berlin.de:2020/all
http://dig.csail.mit.edu/2005/ajar/release/tabulator/0.7/tab.html

URI: http://www3.wiwiss.fu-berlin.de:2020/all        Add to outliner  Help About
This is release 0.7 of the Tabulator Project. The live development trunk is also available.

▶ The Tabulator Project
▶ Decentralised Information Group
▶ W3C (large: includes all specs and groups)
▶ W3C groups (Member-only data)
▶ Life Sciences demo data: GSK3-Beta Drug Target

▼D2R Server contents
label     D2R Server contents
seeAlso
  ▼List of all instances: Conferences
  mentions ▶ Conference
  label    List of all instances: Conferences
  seeAlso  ▶ D2R Server contents
    ▼International Semantic Web Conference 2002
    location    Sardinia
    date        June 9-12, 2002
    type        ▶ Conference
    label       International Semantic Web Conference 2002
    seeAlso     ▶ List of all instances: Conferences
    startDate   2002-10-09T00:00:00.0
    is
    conference  ▶ Trusting Information Sources One Citizen at a Time
    of          ▶ Automatic Generation of Java/SQL based Inference Engines from RDF Schema and RuleML
      ▼Three Implementations of SquishQL, a Simple RDF Query Language
      conference ▶ International Semantic Web Conference 2002
      creator
        ▼Andy Seaborne
        address            Hewlett-Packard Laboratories, Bristol, BS34 8QZ, UK
        has_affiliation    ▶ Hewlett-Packard Laboratories, Bristol
        research_interests ▶ Semantic Web
        type               ▶ Researcher
                           ▶ Person
        label              Andy Seaborne
        seeAlso            ▶ List of all instances: Persons
        homepage           ▶ http://www-uk.hpl.hp.com/people/afs/
Done
```

Figure 2: A Tabulator session showing ISWC data being browsed.

graph using the above described interaction, and can select a pattern of resources, which is then translated into a SPARQL query, the results of which can be *tabulated* onto a table, onto a google map (in the case that the results have geographic location information), and onto an calendar (in the case that the results have calendar information). The user can also edit the SPARQL query, if they wish to create a more complex query than the interface can automatically generate.

The Tabulator is a very low-level tool, in that it presents a tree of the RDF that resides at a URI, and then creates SPARQL queries across that data to show a flattened form. This means that while the interface is ideal for Semantic Web-savvy users that understand the concepts of URI, RDF, data graphs and SPARQL, it is not abstract enough for anybody else. Tabulator also suffers scalability issues due to its implementation as a local SPARQL triplestore inside of a browser (using JavaScript) and as a FireFox extension. This limits the performance, to the point where requesting a large RDF file, or one which has a lot of outward links can cause the browser to freeze and/or crash. If there was a mechanism to resolve small versions of URIs, or if there were better guidelines against publishing files, this might not happen. Alternatively, a technical solution could be employed, offering chunked access to a resource, either at the server-side or client-side, or perhaps by a semantically aware proxy.

4.2 /facet (slashfacet)

The */facet* interface approaches faceted semantic browsing by revealing the semantic structure of the data to the user. The interface first exposes all *rdf:type*s to the user, and allows them to choose one to begin the exploration. Once the user has chosen a type, all instances of that type are displayed, with a facet column created in the interface for all properties of that type. An example interaction where the user has filtered on Claude Monet is shown in Figure 3. Clearly the implications on the forms of the data are significant, and from a user experience point-of-view at least, many assumptions are made about suitability of browsing data in this fashion. The browsing interface employed by */facet* is not as “raw” as an RDF graph walking tool, as by use of its faceted approach it is effectively pre-emptively loading labels for instances across the whole graph and presenting them to the user. This means that the user gets an overview of the space, as in other faceted browsers, while also directly accessing the semantic structure of RDF data.

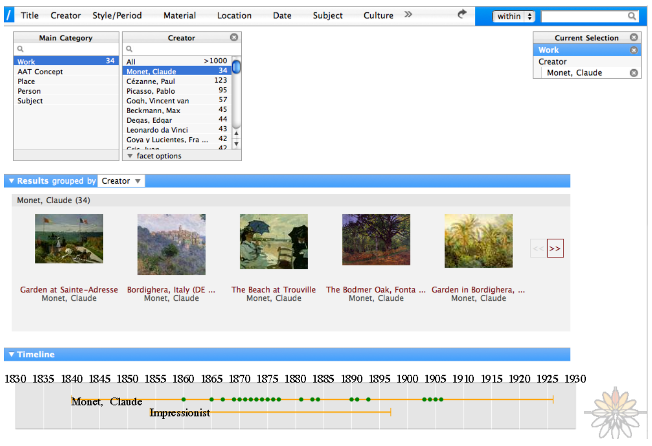


Figure 3: A /facet browsing session showing Claude Monet being browsed.

A downside to this level of abstraction is that users have to understand (or at least navigate) the hierarchies of classes, which can be unobvious, as they would have been modelled for integrity of knowledge, which is not always intuitive. As described in Section 2.3, an abstraction over the supplied RDF could be created using OWL inference and mapping, and RIF rules, enabling a simpler, flattened view to be browsed in */facet*, although this would require extra configuration, more in the vein of mSpace.

The */facet* interface supports the querying knowledge bases using SWI-Prolog, and does not work with RDF files until they have been asserted. The primary */facet* demo runs with over 6 million triples, something that would be unrealistic to expect of a browser to do by realtime querying of resolvable URIs, or RDF files.

4.3 mSpace

mSpace is a multi-faceted browser that has supports a number of backends[18]. In mSpace, the facets are preconfigured by hand, in what is called the mSpace Model. The model describes a first-order “goal facet” (by its *rdf:type*) as one facet, and then describes the rest of the facets via their relationship to the goal facet. From this description, the interface can create facets over a knowledge base and display them as a set of columns, as shown in Figure 4, which also shows an additional feature of the faceted columns, namely backwards highlighting. Backwards highlighting is an interface technique that shows the possible selections in the facets that correspond to the selections that a user makes. In mSpace this is implemented by highlighting all corresponding items to the left of the facet the user has clicked in, and then performing traditional filtering of items to the right of the facet the user has clicked in. By allowing the user to alter the ordering of the columns, a lot of information can be shown to the user in a relatively small amount of screen space.

The penalty for providing this level of rich interaction is in query time. The complexity and number of queries required to expose the backward highlights and forward filtering in many columns is relatively high, and as such, performing realtime queries on remote RDF files or resolvable URIs at realtime would be unrealistic, as with */facet*. As mentioned above, the mSpace interface supports a number of backends, and can be configured to use a traditional relational database if required, instead of querying a triplestore using SPARQL.

5. BEST OF BOTH WORLDS EXPLORATION

Research into browser development has resulted in two different approaches for browsing data on or from the Semantic Web. The first approach is to directly browse the linked data structure of the web, whereby a user inputs a URI as a starting point, and the interface exposes links in that document to other resources on the Semantic Web, allowing the user to browse those. Browsers such as the Disco Hyperdata Browser[7] and the Tabulator[2] take this approach, which benefits from browsing the Semantic Web directly, much like browsing the World Wide Web, in that no centralised server is required The second approach is to harvest data from a variety of sources into a single storage engine, and query the combined data set. This approach means that the knowledge can be optimised for the specific browsing interface, as indexes can be pre-calculated ahead of time.

A model that we propose is a mixed-source browsing interface, whereby an exploratory browser can support querying both pre-processed data, and raw RDF or SPARQL data on the Semantic Web, and adding found data to a processing queue, for faster indexed access in the future. Such a system would combine the benefits of using data that has been optimised for browsing, with the ability to explore live data on the Semantic Web.

In order to fulfil the requirement of data being as up-to-date as possible, the querying infrastructure that supplies data to the interface periodically updates data sources in the background, and re-imports them into the data storage engine, re-indexing the data as it is imported. This update and import process is much that like of a world wide web search engine, although our approach differs in that data is indexed against faceted interface definitions, and not against keywords.

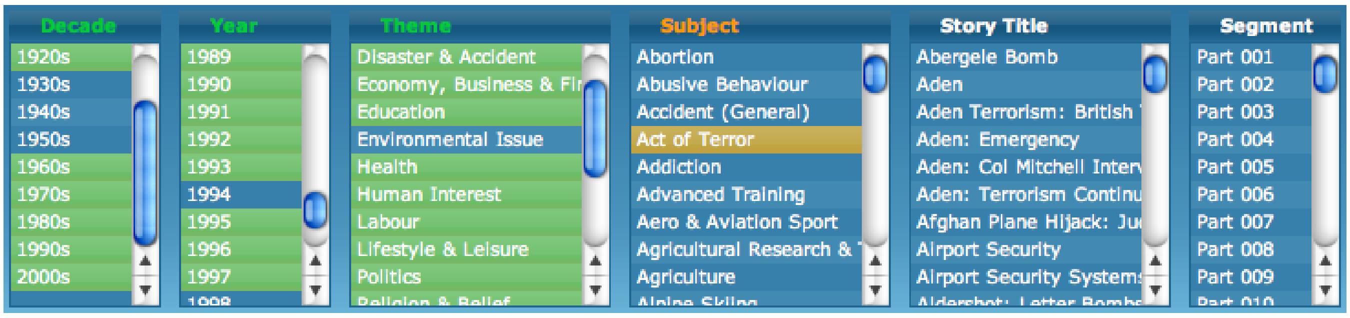
In order to ensure the efficiency of the system, we have utilised Semantic Web tools and techniques that rate the most efficient according to our analysis, as described in Section 3.

.. How to most effectively use the tools shown in the diagram to enable live and cached browsing, and how the facet ontology should work with these tools ..

5.1 Defining a faceted abstraction

One of the challenges faced when providing a data explorer is how to abstract the data so that a browsing interface over the data is understandable, not overwhelming, and enables the user to access the information that they require. When data from multiple sources is combined in an arbitrary way, data covering different domains is represented, and dis-

[7]Disco `http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/`



Decade	Year	Theme	Subject	Story Title	Segment
1920s	1989	Disaster & Accident	Abortion	Abergele Bomb	Part 001
1930s	1990	Economy, Business & Fir	Abusive Behaviour	Aden	Part 002
1940s	1991	Education	Accident (General)	Aden Terrorism: British '	Part 003
1950s	1992	Environmental Issue	Act of Terror	Aden: Emergency	Part 004
1960s	1993	Health	Addiction	Aden: Col Mitchell Interv	Part 005
1970s	1994	Human Interest	Advanced Training	Aden: Terrorism Continu	Part 006
1980s	1995	Labour	Aero & Aviation Sport	Afghan Plane Hijack: Ju	Part 007
1990s	1996	Lifestyle & Leisure	Agricultural Research & '	Airport Security	Part 008
2000s	1997	Politics	Agriculture	Airport Security System:	Part 009
	1998	Religion & Belief	Alpine Skiing	Aldershot: Letter Bombs	Part 010

Figure 4: The mSpace Facet Columns, showing a selection in the Subject column, and many highlights of related items to the left of the selection, and filtered items to the right of the selection.

playing unrelated data together rarely provides a suitable browsing experience: there is too much information to sort through, it is difficult to show the links between the data, and it is harder to determine which attributes of the data are of interest to the user.

There are a number of ways in which this problem could be solved. The browser could limit the data to a particular source, showing only data from that source at one time, while retaining the ability to browse alternate sources when links to data point to them. This approach is the default methodology utilised by semantic web graph walking tools such as Disco and the Tabulator, as they display all information from a supplied URI, and allow walking to alternate data sources by following links within the data. The problem with doing this is that it does not make full use of the available linked data, as data about individual subjects becomes partitioned purely by who supplies the data.

Another way to limit the data that is shown, is by the ontology used. This method appears to be reasonable, as an ontology typically represents a single domain, and data from different providers can use the same ontology. This approach benefits from enabling data from multiple sources to be combined and shown together, while limiting the displayed data to what a single ontology, and therefore a single domain. The problem with this approach is that an ontology can cover so much material that the data is still too vast to explore in a single interface. Additionally, a single domain is often covered by multiple ontologies, and as such limiting the displayed data to a single ontology arbitrarily limits the explorable data to the needs of a particular application of the data, and not the needs of the user now.

The OntoGator system [13, 16] provides a view-based faceted search over semantic web data, enabling users to keyword search over terms, and select from facets to make constraints over the results. This approach works well when the user knows what they are looking for: users can select predicates, and search for instances that they wish to constrain on. This enables users to interactively filter attributes that results must match. When the user does not know the possible attributes of data they want to find, or when they wish to explore the metadata to find out more about the domain, view-based searching is not as effective.

Other previous work has utilised card sorting methods [26] to provide faceted search over a large document set. In their work, card sorting by a selection of example users was used to create an ontology to be used as basis for classification of documents, in order to provide facets to filter over those documents. Card sorting is limited to classifying a stable set of documents, and cannot be applied to dynamically changing metadata, as is the aim of our research.

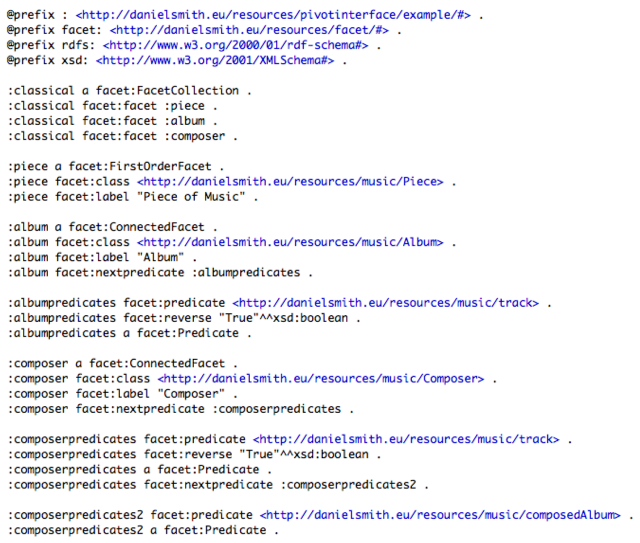
Our approach suggests that in order to enable a scalable browsing experience, the browser should be able to display:

- Multiple different ontologies
- Data from multiple providers

The browser should also be able to do this without overloading:

- The data provider's server
- The client's computer
- The cognitive load of the user

In order to do achieve these, we present FacetOntology. FacetOntology is an ontology that facilitates describing facets, and their connections, from RDF data. The ontology defines three key concepts: facet collections, first-class facets and connected facets. A facet collection is a definition of an interface, which comprises a first-class facet, and a number of connected facets. A first-class facet is defined by its RDFS Class, and connected facets are defined by their RDFS classes, and a chain of predicates that link their instances to those of the first-class facet. For example, consider the domain of Classical Music, with facets of Piece, Composer and Album. In order to define this domain using FacetOntology, a suitable first-class facet must be chosen. A facet is suitable for definition as *first-class* if it is reasonable to suggest that all other facets in this domain relate to that facet more than any other. In this example, we have chosen the Piece facet as first-class, as all Pieces have an Album, and all Pieces have a Composer. To define the Piece facet, we require only to define it using its RDFS Class, see Figure 5. To define the other facets in this collection, we define their RDFS Class, and also a chain of predicates that connects from the first-class facet (Piece) to the connected facet data. For example, Piece and Album are connected with a single predicate `track`. As this predicate is directional from the album to the piece, and our definition is from the first-class facet to the connected facet (from Piece to Album), we must also indicate that the direction of this predicate is reversed, see Figure 5. In order to define the Composer facet, we must define a predicate chain that first joins to Album, and then to Composer, as the ontology that defines the classical music data describes composers as having composed albums, and there is no direct link to the individual pieces. As such, the same predicate definition that is used for the Album facet (see above) is first defined, and then a predicate `composedAlbum` is described, in order to complete the predicate chain.



```
@prefix : <http://danielsmith.eu/resources/pivotinterface/example/#> .
@prefix facet: <http://danielsmith.eu/resources/facet/#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

:classical a facet:FacetCollection .
:classical facet:facet :piece .
:classical facet:facet :album .
:classical facet:facet :composer .

:piece a facet:FirstOrderFacet .
:piece facet:class <http://danielsmith.eu/resources/music/Piece> .
:piece facet:label "Piece of Music" .

:album a facet:ConnectedFacet .
:album facet:class <http://danielsmith.eu/resources/music/Album> .
:album facet:label "Album" .
:album facet:nextpredicate :albumpredicates .

:albumpredicates facet:predicate <http://danielsmith.eu/resources/music/track> .
:albumpredicates facet:reverse "True"^^xsd:boolean .
:albumpredicates a facet:Predicate .

:composer a facet:ConnectedFacet .
:composer facet:class <http://danielsmith.eu/resources/music/Composer> .
:composer facet:label "Composer" .
:composer facet:nextpredicate :composerpredicates .

:composerpredicates facet:predicate <http://danielsmith.eu/resources/music/track> .
:composerpredicates facet:reverse "True"^^xsd:boolean .
:composerpredicates a facet:Predicate .
:composerpredicates facet:nextpredicate :composerpredicates2 .

:composerpredicates2 facet:predicate <http://danielsmith.eu/resources/music/composedAlbum> .
:composerpredicates2 a facet:Predicate .
```

Figure 5: FacetOntology example for Classical Music, in RDF/N3

This facet collection definition allows the faceted browser component of our approach to both identify data that will fit into this collection, and to render it correctly. The definition data is used in two key places: Firstly, the facet definitions are used to generate the queries that select data to be rendered within the faceted browser. The class and predicate definitions are used to generate SPARQL queries that extract data from a triplestore, so that it can be inserted into a database, and indexed, to be browsed by our faceted browser. See Figure 6 for an illustration of the architecture of the data importer. Secondly, the user selects the facet collection, marking it as being of interest. Whenever the user browses a new data source, all facet collections that are marked as being of interest are queried, to see if any data from that source can be rendered using that facet collection.

6. FUTURE WORK

Future work involves implementing a prototype of our approach, and evaluating it against multiple sets of RDF such a DBPedia and Freebase, using various facet descriptions written in FacetOntology. The evaluation is designed to highlight the usefulness of the tool in creating faceted interfaces from large-scale data sources on the Semantic Web, and to demonstrate the effectiveness of abstraction descriptions.

7. CONCLUSIONS

In this paper we have presented our approach for creating dynamic abstractions over data from the Semantic Web, using a scalable and tractable framework. We combine methodologies from Semantic Web browsers that show

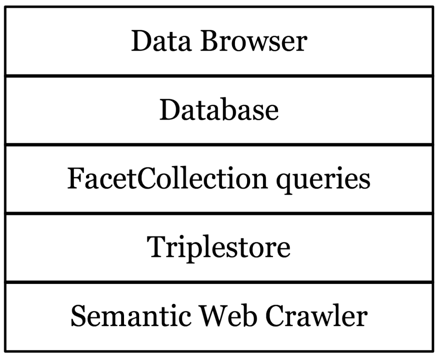


Figure 6: The Data Import stack, showing how data is crawled from the Semantic Web, asserted into a Triplestore, queried from queries generated by FacetCollection definitions, inserted into a database, and rendered by the data browser.

raw RDF structures, and those that use specially crafted pre-gathered subsets of Semantic Web data. In doing so, we resolve the issue of how to support scalable browsing of arbitrary data from the Semantic Web, while giving users the power to select the views over the data that they are interested in.

8. REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. chapter DBpedia: A Nucleus for a Web of Open Data, pages 722–735. 2007. 10.1007/978-3-540-76298-0_52.

[2] T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets. Tabulator: Exploring and Analyzing linked data on the Semantic Web. *Proceedings of the 3rd International Semantic Web User Interaction Workshop*, 2006.

[3] C. Bizer and R. Cyganiak. D2R Server–Publishing Relational Databases on the Semantic Web. *5th International Semantic Web Conference*, 2006.

[4] D. Brickley and L. Miller. FOAF Vocabulary Specification. *Namespace Document*, 3, 2005.

[5] F. Dawson and T. Howes. vCard MIME Directory Profile, 1998.

[6] J. Euzenat. An API for ontology alignment. *Proc. 3rd international semantic web conference, Hiroshima (JP)*, pages 698–712, 2004.

[7] H. Glaser, T. Lewy, I. Millard, and B. Dowling. On coreference and the semantic web, December 2007.

[8] S. Harris and N. Gibbins. 3store: Efficient bulk rdf storage. In *Proceedings of 1st International Workshop on Practical and Scalable Semantic Systems (PSSS'03)*, pages 1–15, 2003.

[9] M. Hearst. Clustering versus faceted categories for information exploration. *Communications of the ACM*, 49(4):59–61, 2006.

[10] M. Hildebrand, J. van Ossenbruggen, and L. Hardman. chapter /facet: A Browser for Heterogeneous Semantic Web Repositories, pages 272–285. 2006. 10.1007/11926078_20.

[11] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From shiq and rdf to owl: the making of a web ontology language. *J. Web Sem.*, 1(1):7–26, 2003.

[12] D. Huynh, R. Miller, and D. Karger. chapter Potluck: Data Mash-Up Tool for Casual Users, pages 239–252. 2007. 10.1007/978-3-540-76298-0_18.

[13] E. Hyvonen, S. Saarela, and K. Viljanen. Ontogator: Combining View-and Ontology-Based Search with Semantic Browsing. *information retrieval*, 16:17.

[14] A. Jaffri, H. Glaser, and I. Millard. Uri disambiguation in the context of linked data. In *Linked Data on the Web (LDOW2008)*, April 2008.

[15] Y. KALFOGLOU and M. SCHORLEMMER. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(01):1–31, 2003.

[16] E. Makela, E. Hyvonen, and S. Saarela. Ontogator-A Semantic View-Based Search Engine Service for Web Applications. *LECTURE NOTES IN COMPUTER SCIENCE*, 4273:847, 2006.

[17] mc schraefel, N. R. Shadbolt, N. Gibbins, S. Harris, and H. Glaser. CS AKTive space: representing computer science in the semantic web. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 384–392, New York, NY, USA, 2004. ACM Press.

[18] m.c. schraefel, M. Wilson, A. Russell, and D. A. Smith. mspace: improving information access to multimedia domains with multimodal exploratory search. *Commun. ACM*, 49(4):47–49, 2006.

[19] D. McRae-Spencer and N. Shadbolt. Also by the same author: Aktiveauthor, a citation graph approach to name disambiguation. In *6th ACM/IEEE-CS Joint Conference on Digital Libraries 2006*, pages 53–55, 2006.

[20] N. Noy and M. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 450–455, 2000.

[21] E. Pietriga, C. Bizer, D. Karger, and R. Lee. chapter Fresnel: A Browser-Independent Presentation Vocabulary for RDF, pages 158–171. 2006. 10.1007/11926078_12.

[22] M. Schorlemmer and Y. Kalfoglou. Progressive ontology alignment for meaning coordination: an information-theoretic foundation. *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 737–744, 2005.

[23] D. A. Smith. Ontobrowse: A world of knowledge. This is the final report for the Part III Project., May 2004.

[24] D. A. Smith, A. Owens, mc schraefel, P. Sinclair, P. André, M. Wilson, A. Russell, K. Martinez, and P. Lewis. Challenges in supporting faceted semantic browsing of multimedia collections. In *The Second International Conference on Semantic and Digital Media Technologies (SAMT2007)*. Springer, 2007.

[25] E. Stoica, M. Hearst, and M. Richardson. Automating Creation of Hierarchical Faceted Metadata Structures. *Proceedings of NAACL HLT*, pages 244–251, 2007.

[26] O. Suominen, K. Viljanen, and E. Hyvonen. User-Centric Faceted Search for Semantic Portals. *Semantic Web Research and Applications*, 2007.

[27] G. Wagner, A. Giurca, and S. Lukichev. A Usable Interchange Format for Rich Syntax Rules Integrating OCL, RuleML and SWRL. *Proc. of WSh. Reasoning on the Web*, 2006.

[28] J. Wielemaker, G. Schreiber, and B. Wielinga. Prolog-Based Infrastructure for RDF: Scalability and Performance. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 644–658, 2003.