UNIVERSITY OF SOUTHAMPTON
Faculty of Physical and Applied Sciences
Electronics and Computer Science

Exploratory and Faceted Browsing, over Heterogeneous and Cross-Domain Data Sources

by Daniel Alexander Smith

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

Supervisor: dr mc schraefel
Examiners: Professor Nigel Shadbolt and Professor James Hendler

June 2011

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL AND APPLIED SCIENCES
ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Daniel Alexander Smith

Exploration of heterogeneous data sources increases the value of information by allowing users to answer questions through exploration across multiple sources; Users can use information that has been posted across the Web to answer questions and learn about new domains. We have conducted research that lowers the interrogation time of faceted data, by combining related information from different sources. The work contributes methodologies in combining heterogenous sources, and how to deliver that data to a user interface scalably, with enough performance to support rapid interrogation of the knowledge by the user. The work also contributes how to combine linked data sources so that users can create faceted browsers that target the information facets of their needs. The work is grounded and proven in a number of experiments and test cases that study the contributions in domain research work.

Contents

List of Figures

Acknowledgements

I would like to acknowledge the direction and input of my supervisor, dr mc schraefel, which began before I formally worked on this research, and has continued after this thesis was completed. I would like to thank Professor Nigel Shadbolt and Professor James Hendler for their input during my examination, which helped me to improve this document significantly. Finally, I am indebted to my family and to Heather for their support.

Academic Thesis: Declaration Of Authorship

I, Daniel Alexander Smith declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

Exploratory and Faceted Browsing, over Heterogeneous and Cross-Domain Data Sources

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;

2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;

3. Where I have consulted the published work of others, this is always clearly attributed;

4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;

5. I have acknowledged all main sources of help;

6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;

7. Either none of this work has been published before submission, or parts of this work have been published as:

Smith, D. A., Popov, I. and schraefel, mc. (2010) Data Picking Linked Data: Enabling Users to create Faceted Browsers. In: Web Science Conference 2010, 26-27 April, 2010, Raleigh, NC, USA. (Submitted)

Smith, D. A., Bretherton, D., Lambert, J. and schraefel, mc. (2010) The MusicNet Composer URI Project. In: UK e-Science All Hands Meeting 2010, September 2010, Cardiff. (In Press)

Smith, D. A., Lambert, J., schraefel, mc and Bretherton, D. (2010) QWIC: Performance Heuristics for Large Scale Exploratory User Interfaces. In: User Interface Software and Technology 2010, October 3-6, 2010, New York, NY, USA. (In Press)

Bretherton, D., Smith, D. A., schraefel, mc, Everist, M., Brooks, J., Polfreman, R. and Lambert, J. (2009) Discovery and exploration using musicSpace. In: Unlocking Audio 2: Connecting with Listeners, 16-17 Mar 2009, British Library.

Bretherton, D., Smith, D. A., schraefel, mc, Polfreman, R., Everist, M., Brooks, J. and Lambert, J. (2009) Integrating musicology's heterogeneous data sources for better exploration. In: 10th International Society for Music Information Retrieval Conference, 26-30 October 2009, Kobe, Japan.

Bretherton, D., Smith, D. A., schraefel, mc, Everist, M., Brooks, J., Polfreman, R. and Lambert, J. (2009) musicSpace: Integrating Musicology's Heterogeneous Data Sources. In: InterFace 2009: 1st International Symposium for Humanities and Technology, 9-10 July 2009, Southampton, UK.

Bretherton, D., Smith, D. A., schraefel, mc, Polfreman, R., Everist, M., Brooks, J. and Lambert, J. (2009) Orchestrating musical (meta)data to better address the real-world search queries of musicologists. In: 5th IEEE International Conference on e-Science.

Smith, D. A., Bretherton, D., schraefel, mc, Polfreman, R., Everist, M., Brooks, J. and Lambert, J. (2009) Using Pivots To Explore Heterogeneous Collections: A Case Study in Musicology. In: All Hands Meeting 2009, 7-9 December 2009, Oxford.

André, P., schraefel, mc, Wilson, M. L. and Smith, D. A. (2008) The Metadata is the Message. In: Web Science Workshop at WWW'08.

Smith, D. A., Lambert, J. and schraefel, mc. (2008) Rich Tags: Cross-Repository Browsing. In: Open Repositories Conference 2008 (OR2008), 1st-4th April 2008, Southampton, United Kingdom. (In Press)

Smith, D. A., Owens, A., schraefel, mc, Sinclair, P., André, P., Wilson, M., Russell, A., Martinez, K. and Lewis, P. (2007) Challenges in Supporting Faceted Semantic Browsing of Multimedia Collections. In: The Second International Conference on Semantic and Digital Media Technologies (SAMT2007), 5-7 December 2007, Genova, Italy. pp. 280-283.

André, P., Wilson, M. L., Russell, A., Smith, D. A., Owens, A. and schraefel, mc. (2007) Continuum: designing timelines for hierarchies, relationships and scale. In: UIST2007 (ACM Symposium on User Interface Software and Technology), Newport, Rhode Island. pp. 101-110.

André, P., Wilson, M. L., Owens, A. and Smith, D. A. (2007) Journey planning based on user needs. In: CHI '07 extended abstracts, April 2007, San Jose, CA. pp. 2025-2030.

Smith, D. A., Lambert, J. and schraefel, mc. (2007) Richtags: Cross Repository Browsing. In: JISC Preservation and Digital Repositories Workshop, November 28, 2007, Bristol, UK.

Wilson, M. L., Smith, D. A., Russell, A. and schraefel, mc. (2006) mSpace Mobile: a UI Gestalt to Support On-the-Go Info-Interaction. In: the SIGCHI Conference on Human Factors in Computing Systems (CHI06), 22-27 April, 2006, Montreal, Canada. pp. 247-250.

Wilson, M. L., Russell, A., Smith, D. A. and schraefel, mc. (2006) mSpace Mobile: Exploring Support for Mobile Tasks. In: The 20th BCS HCI Group conference in co-operation with ACM (HCI06), 11-15 September 2006, Queen Mary, London. pp. 193-202.

schraefel, mc, Wilson, M. L., Russell, A. and Smith, D. A. (2006) mSpace: improving information access to multimedia domains with multimodal exploratory search. Communications of the ACM, 49 (4). pp. 47-49.

schraefel, mc, Smith, D. A., Owens, A., Russell, A., Harris, C. and Wilson, M. L. (2005) The evolving mSpace platform: leveraging the Semantic Web on the Trail of the Memex. In: Hypertext, 2005, Sept 6-9 2005, Salzburg. pp. 174-183.

schraefel, mc, Smith, D. A., Russell, A., Owens, A., Harris, C. and Wilson, M. L. (2005) The mSpace Classical Music Explorer: Improving Access to Classical Music for Real People. In: V MUSICNETWORK OPEN WORKSHOP: Integration of Music in Multimedia Applications, July 4 and 5, 2005, Vienna, Austria.

Wilson, M. L., Russell, A., Smith, D. A., Owens, A. and schraefel, mc. (2005) mSpace Mobile: A Mobile Application for the Semantic Web. In: End User Semantic Web Workshop, ISWC2005, 07/11/2005, Galway, Ireland.

Signed:

. . . .

Date:

. . . .

Chapter 1

Introduction

In this thesis, we first detail our aims and list the requirements in order to meet them. Then second, we identify three fundamental problems that need to be addressed when exploring heterogeneous data sets. Third, we present our system's engineering which is designed to overcome the identified problems. Lastly, we validate the performance of our system and analyse the efficacy of our solutions.

We begin by describing the state-of-the-art in Semantic Web and User Interfaces research, and how the problems in that area inform a requirements analysis for our research.

1.1 Interfaces in the Semantic Web

The goal in the design of the semantic web initially focused on designing a data web where machine reading of information would be easy, and presentation of results to humans fairly straightforward. User Interfaces (UIs) were therefore not present on the original Semantic Web layer cake, shown in Figure 1.1.

The past decade of Semantic Web research and development, and rise in the past 4 years of Linked Data (since Tim Berners-Lee published his rules for Linked Data[1] have shown that humans are working directly with the data, and that consequently, UIs are very much a key part of supporting information discovery, exploration, interrogation, annotation and presentation. This point was formally recognised in 2006, with

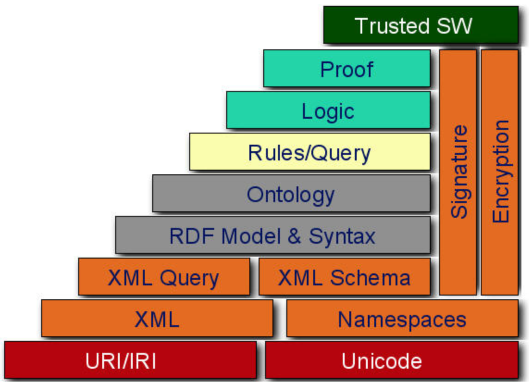


FIGURE 1.1: Semantic Web Layer Cake, circa 2004.

[1]Linked Data, Tim Berners-Lee, 2006-07-27
`http://www.w3.org/DesignIssues/LinkedData`

a revised Semantic Web layer cake, shown
in Figure 1.2.

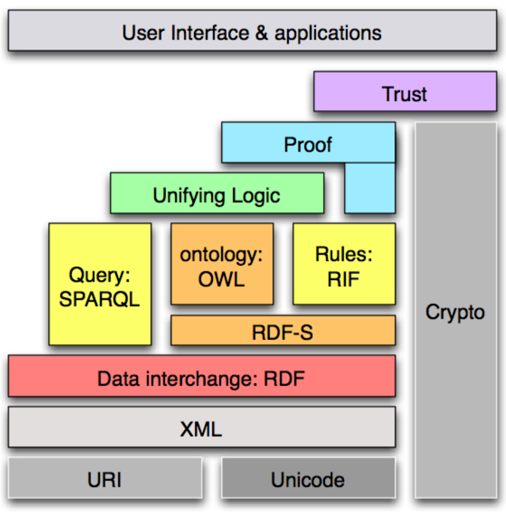


FIGURE 1.2: Semantic Web Layer Cake, circa 2006, which added User Interfaces and Applications.

In the following overview we look at why UIs for the Semantic Web are non-trivial. In this thesis the challenges for interaction design are considered through the lens of a motivating scenario: someone who is not a domain expert, so does not have the terminology of a domain available to aid exploration, and yet who wants to discover information about a domain. This scenario and its rationale are detailed below. From this scenario, five research issues have been developed that have guideed the research for this dissertation. Specifically:

1. Integrity over a Graph of Metadata
2. Entity Co-reference and Multi-source Alignment
3. Performance of an Interface
4. Traversal of Domain
5. User-creation of Interfaces

These research challenges are detailed in Section 1.3. How their solutions are contributions to the state of the art are also presented.

1.2 Overview of the Semantic Web

The vision of the Semantic Web is described by Berners-Lee, Hendler and Lasilla in Scientific American (Berners-Lee et al., 2001) as the concept and standards base which states that when data is published in a structured, knowledge-rich and formal manner, it is possible for machines to: parse and understand the knowledge in that data; infer further knowledge; and automatically merge and combine heterogeneous data sources. This vision is motivated by the following example: a brother and sister, Pete and Lucy, are arranging to share driving duties for their mother's physical therapy sessions over the phone. They instruct their Semantic Web agents to gather a list of health care providers within a radius of their mother's home that are covered under her health insurance plan. Through the affordances of the knowledge on the Semantic Web and the ability for machines to understand any knowledge they encounter, an answer can be found.

In order for the Semantic Web to be able to provide answers to users and allow software agents to manipulate services, a number of different features are defined by the Semantic Web. One of those features which is of key interest to this thesis is the design and standardisation of how metadata is published to the Semantic Web, which is also known as *Linked Data*. Linked Data is both a concept and implementation standard (also known as *httpRange-14*, a reference to its TAG[2] issue name), that describes how to follow data links over the web, in a similar way that humans follow hyperlinks over the web. Metadata on the Semantic Web is represented using the Resource Description Framework (RDF), which presents a logical graph structure to model relationships in metadata. Through the formal and logical structure of RDF, and the soundness of Linked Data's technical architecture, it is now possible to publish semantic and linked data online, which addresses the fundamental protocol stage of enabling the Semantic Web into realisation.

To move further towards the motivating scenario presenting in the Scientific American article there are a number of unresolved challenges to address:

1. Capability for many more systems to output RDF and to communicate changes using web services, so that agents can assert changes and actions, such as altering calendars, as in the Scientific American example. If a system can't *talk Semantic Web*, it can't be involved in this process.

2. Interoperability between heterogeneous systems is a time consuming process. For example, if all of the major supermarkets were to make their data available as RDF, the use of that data is limited without their outputs sharing concepts and attributes so that they can be directly compared.

3. Scalability of the use of RDF data sources. As the amount of potentially relevant knowledge in the world increases, and depending on the aim of your software or agent, large amounts of data might need to be parsed, evaluated, stored and queried.

[2]W3C Technical Architecture Group (TAG): `http://www.w3.org/2001/tag/`

In addition to these core problems, is the challenge of what the user interface looks like, to answer the questions in the scenario: the Scientific American article suggests software agents that will alter calendars, but does not cover how the users themselves input the questions or explore the information and potential results. There is therefore a challenge of how to get this information to the user in a reasonable way, so that it makes sense to them, and enables them to understand the scenario and the information they are looking at.

Early work assumed that the Semantic Web would be used primarily by software agents, and thus the early focus of the use of the knowledge on the Semantic Web was not on UIs for humans, but on logic and representation issues for agents. When agents can give you answers you need, and solve problems for you, it may be unnecessary for the user to deal directly with the data, however, given the absence of an agent to perform a required task, or when the user does not have the required knowledge of a domain to inform the agent what they need, some UI to allow them to explore and build knowledge of a domain will be useful and required.

The challenge therefore is how to make the knowledge available to users. One approach is to recognise that while the world wide web is a web of documents, the Semantic Web is a web of metadata. For the web, information retrieval techniques create ranked lists of documents that are related to query keywords by examining the contents of those documents, and the pages that link to them. With metadata on the Semantic Web we have a richer graph of knowledge to work with, and it stands to reason that retrieval and exploration of this data can also be made richer, through the use of the metadata.

The provision of a user interface onto the Semantic Web was not an initial focus of Semantic Web research. The 2001 Scientific American article (Berners-Lee et al., 2001) focuses on software agent interaction on the Semantic Web. To the best of our knowledge, the first published work on how a Semantic Web interface might work was published in 2004 (Quan and Karger, 2004). In addition, 2004 also saw the first Semantic Web User Interfaces workshop at the World Wide Web conference in New York. Further to this, mSpace's faceted approach, using Semantic Web data was published in 2005 (m. c. schraefel et al., 2005). One year later in 2006, the W3C's Semantic Web Layer Cake[3] was modified to include User Interfaces and Applications, thus making them officially recognised as a key focus of research into the Semantic Web.

Thus, tools to browse simple Linked Data have emerged, and enable people to search for information on their interest domain, although people still need to know what questions to ask. This seems obvious, however it is more complex than it seems. In some domains it is not intuitive, for example Classical Music, which is a topic where people may not have the knowledge of specific terminology to use in questions, but can recognise different

[3]The W3C Semantic Web layer cake, which outlines the interfaces between technologies in the Semantic Web: `http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/#(24)`

motifs in audio and do not know what they are called.

Furthermore, such tools are limited to not being able to process all data on the Semantic Web in realtime, and thus, our focus remains on producing an interface that delivers responsiveness, at the expense of not being able to explore the whole Semantic Web at large. An approach might apply faceted browsing to individual domains of information that are available on the Semantic Web. The benefit of such an approach is that ensures a focused browsing experience within a particular domain. We also wish to enable more heterogeneous browsing, by enabling users to move between the individual faceted domains where overlaps in the domains existed, even if those overlaps are not yet documented.

We continue by describing the three core problems we will tackle, which will inform the requirements analysis for our research.

1.3 Three Key Problems

Our requirements come from three main problems that must be addressed in order to meet the aim of the thesis:

1. **How to enable people to explore unfamiliar domains through interface design.**

   An abundance of data is available online, however, data on the web is typically curated to a fixed schema, and used for a specific publised purpose. Specifically, data is presentes as individual websites, with little to no connections that enable data from multiple sites to be queried and filtered at once. Thus, users cannot explore the crossover between sites, unless they already have technical skills in web scraping and interface configuration.

2. **How to give users rich features for browsing without sacrificing the performance that makes them want to use it, through back-end data processing methodologies.**

   With every additional feature that is added to an interface, additional load is placed on the client, server and user, leading to performance issues that degrade the user experience. Thus, a challenge is to deliver additional features without degrading performance.

3. **How to enable users to make connections across data sources through use of Semantic pivoting, a data processing and user interface methodology.**

   Exploratory interfaces add value to user experience by allowing users to break down an information space from the starting point of their choosing, however this fails to exploit the natural linkage between data spaces. Thus, our third challenge is to enable users to explore across data sources using the natural semantic linkages that exist in data.

1.3.1 Problem 1: How to enable people to explore unfamiliar domains.

We situate our research in a domain based model, where users have an interest, and are willing to learn, but do not have much knowledge of the area, and are not sure what is a good starting point. An approach (m.c. schraefel et al., 2003) to informing users unfamiliar with a domain has been to expose facets and attributes of a domain in order to allow them to self-guide their exploration of a domain. The approach used preview cues to allow users to decide what to choose based on a sample of audio, so they could self-direct their exploration. This approach scales to different levels of existing knowledge, for example someone with some prior knowledge of classical music, perhaps from taking piano lessons as a child, may have knowledge of the domain that they can use to guide their initial exploration, whereas someone without that knowledge can use the sounds they hear in the music as an initial guide, and correlating their own understanding of the relationships between the facets of the metadata and the musical audio of the music. Such an approach has been explored in terms of a user interaction approach, which has been called a faceted exploratory search (m.c. schraefel et al., 2006).

Exploratory search (Hearst, 2006) has been shown to be a useful and usable approach to knowledge building, based on the exploration of metadata, and we can apply an exploratory approach to browsing multiple heterogeneous sources, where one source may have audio and another has metadata about classical music. Such as approach combines heterogeneity with a powerful exploratory UI. Linked Data has the potential to aid in music discovery, as it can provide metadata about pieces, recordings, composer and so on. Thus, Linked Data, and more generally, the Semantic Web, might be a useful source of data to enable a metadata-based exploration, for users unfamiliar with that domain. To this aim, we submit that with the mechanisms to drive those sources to a UI, that the solution to aiding users in learning about such domains might be a UI that allows intuitive lightweight exploration of metadata.

Documents on the Semantic Web primarily contain interlinked metadata. Metadata is “data-about-data” which is typically in the form of attribute information, for example a photograph will have metadata such as the width, height, file format, file size and so on. The Semantic Web allows the specification of metadata attributes to be declared in a decentralised fashion, which offers benefits such as being able to declare metadata attributes that have never been used by anyone before. Metadata from the Semantic Web lends itself to facet-based exploratory search, since metadata is essentially a collection of facets about entities, and thus modelling domains sources from Semantic Web data should be reasonable to present through a faceted interface. We can then take advantage of exploratory search on top of the Semantic Web as an interim solution to browsing an entire Semantic Web.

Metadata can be likened in some ways to the index card catalogues used in libraries as a way to find books. A library would typically have a number of sets of physical cards for the whole catalogue, each indexed by different attributes, typically *author*, *subject area* and *title*. These indexes mean that books can be located if only one of the attributes is known — if a library only has a subject index, such as Dewey decimal (Dewey, 1876), a reader needs to know the subject before they can find a book. By also indexing by author and title, a reader can find books based on the knowledge that they already know. For example, one might wish to locate a particular book given only the author and title. Without knowing how the librarian classified it, they would not be able to locate the title, except for asking the librarian themselves. Indexes by author and title mean that the reader does not have to concern themselves with a classification. Of course, computerisation of library catalogues mean that keyword search operations can be performed against a database, without the need for examining the classification at all, if the user has a particular book they need. However, if the user does not know which particular book they want, but instead needs to find a book on a particular subject, the classification is a reasonable place to start. Likewise, they may wish to limit their search for books by other attributes, such as the country of publication, or whether the book is paperback or hardback, although there is a limit to how many attributes can be used as an index when using paper index cards, which isn't the case with computerised records. One of the advantages of having the metadata for records digitally, and structured, is that we are no longer limited to exposing fixed facets, as above, and instead, any facets can be used, to explore books, for example, by their *publisher*, *country of publication* or *language*. These attributes are also called facets, and computer UIs to browse records by facets are known as *faceted browsers* (Allen, 1994). Faceted browsing allows users to use any facet of the available information to limit their search, enabling users to use the facet they are most familiar with, or most interested in, as the basis for their exploration. One benefit of faceted browsing is that users can build knowledge of a domain by exploring the relationships between facets; by selecting *Science Fiction* in Genre, I can quickly see a list of Authors that publish Sci-Fi, for example.

In order to apply the benefits of faceted browsing to any domain, structured metadata from that domain must be available. The Semantic Web offers the infrastructure to publish structured metadata on any subject, and as such, the ultimate challenge is how to apply faceted browsing to the Semantic Web, in order to enable users to explore and build knowledge on any domain.

To apply a faceted interface onto the Semantic Web, a number of challenges are faced:

1. **Gathering Data** Knowledge on the Semantic Web can be published by anyone, hosted on their own servers, anywhere on the internet. Knowledge published on

the Semantic Web can point to any other information elsewhere on the Semantic Web, creating a rich graph of relationships. In order to gather all of the data that is referenced, these relationships on the Semantic Web need to be crawled and explored. This structure enables anyone to publish knowledge relating to anything on the Semantic Web, but it leads to a challenge of how to know what knowledge should be gathered and exposed to users, what knowledge should be prioritised, and how to trust it — after all, anyone can publish anything they like, so if an interface exposed any information, there's no guarantee that it'll be correct.

2. **Scalability** In addition to the above challenge of knowing what to gather, there is also the challenge of how much to gather and expose. The more information is gathered, the more computational resources are required to host and query that information. Likewise, there is limit to how much information can be processed by the user, before *information overload* will slow them down, and negatively impact their experience.

3. **Ontology alignment** One of the benefits of the Semantic Web is that attributes and types of information can be defined and published by anyone, and those definitions can be utilised by anyone else, through the use of ontologies. A major challenge in utilising information from multiple sources is that they may not use the same ontology, since multiple ontologies could model the same domain, and thus the ontologies need to be aligned (Euzenat, 2004).

4. **Entity co-reference** In addition to ontologies requiring alignment, the same problem occurs in terms of references to instances, for example to a particular book. One library system may refer to it using the ISBN number, while the other uses the UPC barcode — on the Semantic Web the problem is similar, although information is referenced by URIs, and problems arise in that two sources may use difference URIs to refer to a resource. The problem is known as entity co-reference, and the challenge is in how to perform disambiguity analysis on entities (Hirschman et al., 1997).

Thus, in order to enable users to explore all useful data in a faceted interface the above problems must be tackled. This does not mean that, for example, ontology alignment must be solved generally for the entire Semantic Web, but that our research must develop some approach to the challenge that it poses, in order to meet our goals.

1.3.2 Problem 2: How to give users rich features for browsing without sacrificing the performance that makes them want to use it.

Applying the affordances of faceted browsing to the Semantic Web are a challenge in terms of addressing performance of the interface across large scale data, and addressing new user interface techniques that can be created, given the novelty of structured data across multiple domains that we can filter at scale. The ability to ask questions of large scales of data opens up new ways to help users build knowledge over that data, and to aid them in exploring that data, which are challenges that surface only when an interface has the ability to filter over large scale data, and deliver that to the user responsively. Research into addressing such challenges of how to display a large number of facets, or how to hint to users that selections are available, but not currently shown, are therefore very much user driven, and our research focuses on addressing challenges in users' interaction with the data.

When dealing with data that is larger than small-scale personal data, problems around performance begin to impact on the user experience. When performance problems cause responsiveness of the interface to dip below an acceptable threshold (for example that the interface doesn't fully update within 2 seconds after clicking), users have been shown to lose engagement and stop using the system. Performance problems can be split into two areas:

- **Query performance on the Back End** The more data that is being queried over, the greater the computational load on the database or triplestore. In order to deliver the performance required to return everything the interface needs to update, in the required timeframe, the computational load needs to be managed, using whatever engineering methodologies required.

- **Client performance on the Front End** When dealing with large amounts of metadata being returned from the server, the load of rendering these can impact performance on the client-side front end of the interface. In particular, there are constraints on the amount of data that can be sent to the user (limited by the bandwidth of the user's internet access) as well as a limit on the amount of data that can be rendered in their web browser before performance is too far impacted. Specifically, if too many elements are sent to modern web browsers, they slow down and get to a state where they are unusable. Thus, in order to support intuitive browsing of hundreds of thousands of items, innovative techniques will need to be developed and evaluated in order to achieve required levels of performance on the client side.

In order to meet our aims, we have to develop an approach that scales to large data, in term of both the front end and the back end.

1.3.3 Problem 3: How to enable users to make connections across data sources.

Four key problems present themselves when dealing with making connections across data sources:

- **Integrity over a Graph of Metadata** How to enforce integrity of the Semantic Web's graph structure for faceted browsing. The Semantic Web's relationships within metadata provide a rich network of metadata, however the connections between attributes are not always necessary to enforce on a per-entity basis, and in other cases are necessary to enforce for the semantic meaning behind the relationships to remain intact. For example, exposing the facets of "Composer" and "Composer Birth Year" in a Classical Music interface, it is important to enforce the integrity between composer and composer birth year, so that when a user selects a particular birth year, only musical works that were composed by composers born on that year are exposed, and also that only composers that were born on those years are exposed. Without the link between composers and their birth years (for example in a flat facet-to-record relationship), any musical work that has multiple composers will erroneously expose all composers for those musical works, where one of the composers has a birth year that was selected.
- **Entity Co-reference and Multi-Source Alignment** When dealing with data from multiple sources, one issue is that references to entities may not share identifiers, so that it is not possible for a system to automatically determine that metadata from multiple sources is referring to the same entities. For example, one classical music source may assert that there is a composer, his name is "Johann Sebastian Bach" and he was born in Eisenach, Germany. Another source may then assert that there is a recording of a work called "Toccata and Fugue in D minor" available as an MP3, by a composer called "J.S. Bach." If these are asserted into a knowledge base, the records will not link up, and therefore a query asking for "All MP3s from composers born in Eisenach" will not yield any results, even though through the above assertions, we know there is at least one. Thus, in order for a faceted browser to display the correct results to the user, a solution to co-reference data that a faceted interface uses needs to be in place so that multiple sources can be aligned, and tuned to a reasonable degree of accuracy. Work to date on co-reference disambiguation is split between approaches that have some heuristic knowledge of a domain, and those which attempt to work over any structured data. For example, it has been shown that authors of scholarly publications tend to cite their own work, and thus a *D.A.Smith* that cites *Daniel Smith* is likely to be the same person (McRae-Spencer and Shadbolt, 2006). Whereas approaches to co-reference over the general Semantic Web can provide possible matches over a large corpus of hetrogeneous data, but a lower degree of certainty (Jaffri et al., 2008).

- **Traversal of Domains** While traditional faceted browsing models are focused on single domains, we must extend this model to support multiple domains, using a technique that allows users to pivot across them, where linked data from both domains exists. For example, there may be linked data about prices and stock of food from an online grocery store, while there is also information on food and nutritional content from a specialist website. The interface should have the capability to enable the user to move between these data sources, using a pivot of the food products, which are the shared facet between both data sets.
- **User-creation of Interfaces** Finally, we require an approach for enabling users to create interfaces themselves, from whatever data sources they wish. The interface should require a minimal amount of knowledge of the challenges and technologies involved. Specifically, we wish to avoid (where possible) exposing RDF graphs to users or segments of SPARQL queries, which is often seen in prototype research interfaces. Part of our approach may therefore develop a mechanism whereby alignment of data sources (see point above) can be attempted in a lightweight way, with automatic publication of the mappings and assertions generated, so that domain experts can examine the results of an automated alignment, and assert which results are correct, and which are incorrect. These results will then be reimported and used in the interface that has been created.

The above problems are developed into a requirements analysis that informs our research, which is detailed in Chapter 3. In the next section we detail the contributions of our work.

1.4 Contributions

The work documented in this thesis has led to the following primary contributions:

1. Data Abstraction Models for Exploring Heterogeneous Data
   (a) mSpace Model
   (b) Facet Ontology

   Our first contribution is a data abstraction model for exploring heterogeneous data sources. Our initial contribution mSpace Model, as outlined in detail in Section 4.2.1, enables a domain to be marked up, so that it can be used by the mSpace faceted explorer. Our research led to a generic and more for contribution of Facet Ontology, as detailed in Section 4.5. Facet Ontology is an OWL ontology that enables any domain to be marked up in generic faceted browser terms so that it can be used by potentially any faceted browsers, rather than just the mSpace interface.

2. Exploratory Interface Performance and Scalability Approaches

    (a) Backend Performance Approaches

    (b) UI Performance Approaches

   Our second contribution is a series of heuristics and approaches for both back-end and front-end performance improvements for exploratory interfaces. In Section 4.4 we overview our contributions of scalable approach to supporting large scale data for faceted browsers. In Section 5.3.5 we detail our contribution of dynamically paging lists that enable UI performance improvements for faceted interfaces.

3. User-creation of faceted interfaces over Semantic Web data

    (a) Data Picker

    (b) Facet Portal

   Our third contribution is an approach to enable users to pick data from RDF and Linked Data sources, and create high-performance faceted interfaces from those sources. In Section 4.5 we detail our data picker UI which enables users to pick data, and have it turned into a scalable faceted browser. We then detail our Facet Portal contibution which enables other faceted browsers to be discovered.

4. Pivoting across faceted browsing interfaces

    (a) Semi-automated alignment for cross-interface faceted browsers: Facet alignment service

    (b) Pivoting Faceted Browsing

   Our fourth and final contribution is an automated alignment service that enables users to browse across faceted interfaces of heterogeneous data sources. We detail our alignment contribution, and pivoted browsing in Section 4.8, which enables two different faceted interfaces to be combined, creating a pivot across a facet from each interface. Users can then view a faceted browser with both datasets, to ask more complex queries.

Overall, these contributions have led to the following peer-reviewed publications:

mSpace: improving information access to multimedia domains with multimodal exploratory search

schraefel, mc., Wilson, M. L., Russell, A. and Smith, D. A. (2006) *mSpace: improving information access to multimedia domains with multimodal exploratory search.* Communications of the ACM, 49 (4). pp. 47-49.

Available at: `http://eprints.ecs.soton.ac.uk/12376/`

In this article, the mSpace faceted browsing interface is detailed. My contribution to the article is the mSpace Model (as described in detail in Section 4.2.1 of this document) which defines how to describe a structured data source so that a faceted browser can be created from it. This article appeared in Communications of the ACM and disseminates Contribution 1.

Challenges in Supporting Faceted Semantic Browsing of Multimedia Collections

> Smith, D. A., Owens, A., schraefel, mc, Sinclair, P., André, P., Wilson, M., Russell, A., Martinez, K. and Lewis, P. (2007) *Challenges in Supporting Faceted Semantic Browsing of Multimedia Collections.* In: The Second International Conference on Semantic and Digital Media Technologies (SAMT2007), 5-7 December 2007, Genova, Italy.

Available at: `http://eprints.ecs.soton.ac.uk/14507/`

In this article I outline the challenges we faced when applying the mSpace faceted browsing framework across a large multimedia collection. The challenges of this paper are explained in detail in Section 4.4 of this document. This paper disseminates Contribution 2.

Using Pivots To Explore Heterogeneous Collections: A Case Study in Musicology

> Smith, D. A., Bretherton, D., schraefel, mc, Polfreman, R., Everist, M., Brooks, J. and Lambert, J. (2009) *Using Pivots To Explore Heterogeneous Collections: A Case Study in Musicology.* In: All Hands Meeting 2009, 7-9 December 2009, Oxford.

Available at: `http://eprints.ecs.soton.ac.uk/17998/`

In this paper we discuss how our pivoting approach (see Section 4.8) works, in the context of musicology. Specifically we outline how using our pivoting approach negates the need to design a unifying ontology before integrating multiple heterogeneous sources, and instead derive one automatically, based on the fields that are chosen to be integrated. This paper disseminates Contribution 4.

Data Picking Linked Data: Enabling Users to create Faceted Browsers

> Smith, D. A., Popov, I. and schraefel, mc. (2010) *Data Picking Linked Data: Enabling Users to create Faceted Browsers.* In: Web Science Conference 2010, 26-27 April, 2010, Raleigh, NC, USA.

Available at: `http://eprints.ecs.soton.ac.uk/20804/`

This paper describes our data picker interface, and its architecture that enables users to pick data from RDF sources and Linked Data in order to quickly generate mSpace faceted browsers over that data. We give UK Government public open data as the context, motivated by using a faceted interface as a means to extract data of interest from a large dataset. We then suggest our approach enables users can utilise their existing rich aggregate operations and statistical analysis tools over the extracted data, such as PivotTable and charting functionality found in spreadsheet applications. This paper disseminates Contribution 3.

1.5 Structure of Thesis

In order to address all of the above challenged, we have analysed the related work and derived our own novel approaches to enabling users to create faceted interfaces across heterogeneous data. Firstly in Chapter 2, we outline the related work in more detail than above, and highlight where gaps in the existing work exist, with regard to addressing our challenges.

In Chapter 3 we develop and outline a requirements analysis for our research, based on the three key problems outlined above. We describe eleven research requirements that, when met, will enable users to be able to create interfaces over multiple datasets. Our requirements outline that those interface enable pivoting across domains, with basic automated alignment, so that users with little technical skill can create semantic pivots where they do not already exist.

In Chapter 4 we present our solutions to the problems, which meet the requirements laid out in the previous chapter. We first present our solution to picking data from the semantic web, called the Data Picker, which allows used to pick data so that it can be viewed in a faceted browser. We continue by describing our mSpace Maker system which is used by the Data Picker to translate the description of the picked data into an mSpace faceted browser. We present the mSpace faceted browser, and specifically the performance research that enabled the browser to add new features, while also retaining high levels of performance and responsiveness. We move on to describe the Facet Portal and our approach for semi-automated semantic alignment across datasets, which enables datasets to be overlapped automatically. We then describe our approach to providing an interface over the overlapped datasets, which we call Semantic Pivoting. We then present our work on mobile on-the-go knowledge building, through the mSpace Mobile research into faceted browsing and mobile sensemaking.

In Chapter 5 we validate our solutions against three criteria:

1. Formal User Studies
2. Tasks Now Made Possible
3. Engineering Tractability

Through these three criteria we are able to demonstrate that our work has been formally validated in peer-reviewed publications, that it enables new tasks to be completed, and that our research has improved the tractability of state-of-the-art engineering in our research area.

In Chapter 6 we summarise our contributions, and outline potential future work to extend our approaches.

Chapter 2

Literature Review

There are a number of challenges faced when enabling users to create interfaces over heterogeneous data (as previously discussed in Chapter 1), and in this chapter we present the state of the art which addresses these challenges. In Section 2.1 we outline the challenges in publishing and finding information online that is intended for human consumption, and the challenges faced enabling users to find information online and enabling users to contribute to the creation of information through *Web2.0* sites like *Wikipedia* (Giles, 2005) that allow users to add and edit page content. Section 2.2 discusses the challenges faced when merging heterogeneous data from multiple sources, specifically regarding schema and ontology alignment and entity co-reference. Section 2.3 discusses the concepts of *lightweight* and *heavyweight* semantics, and how rich semantics can be exploited without enduring the high computational cost of inference over fully formal ontologies. In Sections 2.4 and 2.6 we analyse the challenges and related work in creating exploratory interfaces over textual and faceted data, analysing the existing approaches, and situating our approach. Section 2.7 discusses the challenges in delivering realtime query performance over large data sets to support fast browsing in exploratory interfaces, and we analyse existing approaches. In Section 2.8 we analyse approaches to marking up data for human use, such as hierarchical categorisation, tagging and their relationship with the Semantic Web. Section 2.9 concludes with an analysis of approaches to pivoting across heterogeneous data sources.

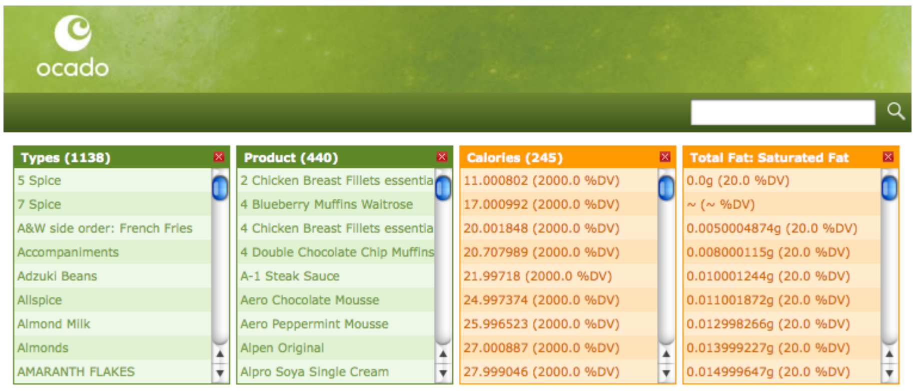
2.1 Browsing Data on the Web

Finding data on the web can be challenging due to the number of documents online. There are various technologies that aim to help both humans and computers to read and analyse data. We discuss in the following two sections how technologies help humans and computers find data.

2.1.1 Technologies that enable Humans to Browse Web Data

Primarily web pages are authored and read by humans, and can be described as a human-to-human activity. Humans focus on publishing web pages that are aesthetically pleasing and are easy to navigate through links, rather than enabling machines to easily understand and process the web page's content. Pin pointing specific data on pages can be challenging because of the lack of semantics. By far the most common method for finding data on the web is the search engine which has pioneered new search capabilities for specialised domains, such as images, maps, and even finding items to buy.

When terms of interest are known to the user, keyword searching can serve as a quick way to filter documents of interest, but limiting by matches to an index. However, when terms of interest are unknown, other classification-based techniques can be utilised. One such technique is faceted browsing, which offers users a number of facets that documents or items of interest have, and the range of values to which those facets can be. For example, an online shopping website may offer the facet of *price*, allowing the user to select a range of acceptable prices, or the facet of *Department*, offering the user to filter from *Produce*, *Frozen Food*, *Bakery* and so on. By showing the user the set of possible values in the whole data set, it allows them to get a picture of the make-up of the domain, in this case groceries, very quickly, without any prior knowledge. In Figure 2.1, mSpace (m. c. schraefel et al., 2005), a faceted browser, is shown in its initial state running over a set of grocery store data. Various facets are shown, and the user is free to select any values to get results.



Types (1138)	Product (440)	Calories (245)	Total Fat: Saturated Fat
5 Spice	2 Chicken Breast Fillets essentia	11.000802 (2000.0 %DV)	0.0g (20.0 %DV)
7 Spice	4 Blueberry Muffins Waitrose	17.000992 (2000.0 %DV)	~ (~ %DV)
A&W side order: French Fries	4 Chicken Breast Fillets essentia	20.001848 (2000.0 %DV)	0.0050004874g (20.0 %DV)
Accompaniments	4 Double Chocolate Chip Muffins	20.707989 (2000.0 %DV)	0.008000115g (20.0 %DV)
Adzuki Beans	A-1 Steak Sauce	21.99718 (2000.0 %DV)	0.010001244g (20.0 %DV)
Allspice	Aero Chocolate Mousse	24.997374 (2000.0 %DV)	0.011001872g (20.0 %DV)
Almond Milk	Aero Peppermint Mousse	25.996523 (2000.0 %DV)	0.012998266g (20.0 %DV)
Almonds	Alpen Original	27.000887 (2000.0 %DV)	0.013999227g (20.0 %DV)
AMARANTH FLAKES	Alpro Soya Single Cream	27.999046 (2000.0 %DV)	0.014999647g (20.0 %DV)

FIGURE 2.1: Screenshot of a mSpace Ocado initial slice, showing four facets, two from Ocado and two from NutritionData.com

Further to retrieving results, another benefit of faceted browsing is that the possible values of faces are also filtered during a faceted search session, so, for example, when a users selects *Bakery* from the *Department* facet, the *Storage* facet then only shows results from the *Bakery*, omitting those which do not result in *Bakery* products (such as *Frozen* or *Tinned*). In Figure 2.2, the user has selected the *Bakery* department, and the possible selections from the other facets has been filtered.

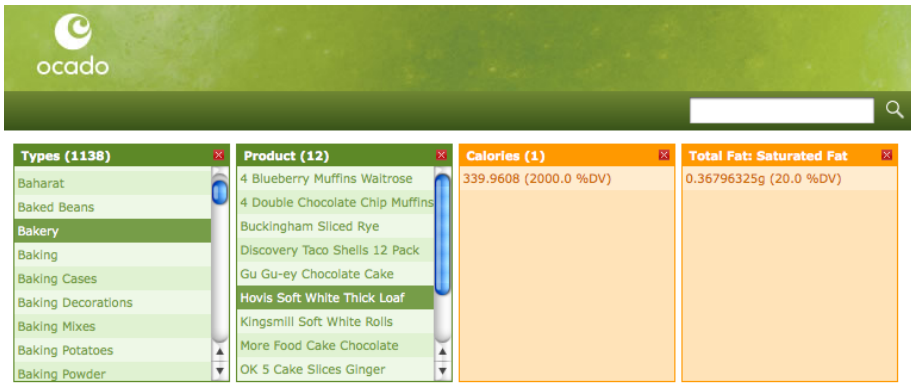


FIGURE 2.2: Screenshot of a mSpace Ocado initial slice, showing four facets, two from Ocado and two from NutritionData.com. A user had selected Bakery, and Hovis Soft White Thick Loaf.

In order to create the best possible search interface, modern interfaces utilise technologies such as AJAX or Flash in order to provide fast response times and to user actions. Such technologies power the so-called *Web2.0* (O'Reilly, 2007), which describes the evolution of the Web into a public forum where traditional media, user-created content and social networks are available to users through interfaces where they can comment, tag and share on pages and content of interest.

Web2.0 has also led to a rise in user-created-content (Ondrejka, 2004), such as blogs and videos. Due to this content being created by users, the demand has been that users can share their own content over multiple sites, by either embedding content such as videos or by syndicating their blog posts. In order to enable data from user created content such as messages, videos, blog posts to be shared over multiple sites, APIs and mechanisms to share data of these different types across the world wide web are now in popular use, leading to a rise in the creation and availability of large volumes of heterogeneous data.

A challenge when dealing with heterogenous data is how to refer to data from different sources, which are likely to use various different data models and serialisations. This is a challenge for our research because the kind of connectedness that enables exploratory browsing to be richer than keyword document search relies upon the system being able to tell when concepts referenced in one data source are equivalent to those in another data source, so that, for example, a site that lists people that *work with* each other can be recognised as an equivalent relationship to another site that lists *colleagues*.

In particular, data models include schemas, where a schema is a data model that describes expressible entities, attributes and their relationships; and ontologies, formally defined as "an explicit specification of a conceptualization" (Gruber, 1995), an ontology is an extension of a schema which details the formal semantics of entities, attributes and relationships. Data can be encoded using different serialisations, where a serialisation details how the structure of a the data model is written to a file (or, more generically a byte-stream), in order to be saved, shared and re-loaded. Heterogeneous data is likely to have been sourced from multiple repositories, collections or databases. Therefore, in order to use heterogeneous data from multiple sources for a single purpose, the data is typically aggregated and pre-processed into a common format, and the correspondence between its referenced entities and concepts determined, which is known as ontology alignment (Euzenat, 2004), and has its basis in work on database schema alignment (Batini et al., 1986).

In order to enable computers to programmatically process data that is shared on the world wide web, a concept and standardisation programme called the Semantic Web was created. In the next section, we overview the technology used so that web pages are machine readable.

2.1.2 Technologies that enable Computers to Use Web Data

The web contains heterogeneous data and in order for that data to be understood and processed by computers must be defined. Heterogeneous data can, by definition take a number of different forms. A high level differentiator of heterogeneity is the amount of structure that the data contains. On one end of the scale (see Figure 2.3), is free text, which contains little structure above that of the language it is written in, while on the other end of the scale data written in RDF and corresponding to an OWL ontology is made up of a specific graph structure with a rich semantic formal knowledge model.

The Resource Description Framework (RDF) is a metadata model for describing a distributed data graph, and is a core component in the Semantic Web. RDF allows statements to be made about entities, in order for them to be published and shared over the world wide web. RDF uses Universal Resource Identifiers (URIs) to identify entities and properties, using "triples": statements made using "subject," "predicate," "object" notation, where each are identified using URIs. The formal semantics of RDF metadata are specified in ontologies, using a relevant ontology language, such as RDF Schema (RDF(S)) or the Web Ontology Language (OWL). An ontology specifies a namespaced collection of class hierarchies and related predicates, and a reasoner can be used to perform inference over RDF data, in order to make entailments over the metadata.

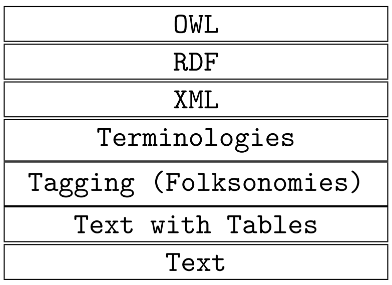


FIGURE 2.3: Scale of Data Heterogeneity

The data scale noted in Figure 2.3 is broad and one-dimensional, and while it captures a number of key data formats, it fails to capture a number of aspects of data that are encountered, and which increase the heterogeneity of data, and the level of work required to normalise such data to a common format. Such aspects include:

Serialisation of data Which describes how the structure of the data model has been written to a file, in order to be shared amongst different tools. Extensible Markup Language (XML) is an example of a popular serialisation that provides a nested hierarchy, namespacing, definable character encoding and entity attributes. *YAML Ain't Markup Language* (YAML) is an example of a serialisation format that provides a human readable format for serialising hierarchies, hashes, relational tree and class definitions.

Entity reference format Different data formats can refer to specific entities in a number of different ways. Resource Description Framework (RDF) uses Universal Resource Identifiers (URIs), to provide a globally resolvable and DNS-reliant globally namespaced identifier, a warehouse system may use the Universal Product Code (UPC) of a product as the identifier, and a departmental database might use a person's full name to identify them.

Model and Schema Knowledge can be modelled in many different ways. Semantic Web systems typically utilise a subsumption architecture, while other systems such as Topic Maps utilise a topic-centric model. RDF can theoretically use any ontology language for its formal semantics, and typically uses RDF(S) or OWL for its formal modelling. In order to align heterogeneous data, one must be familiar with the ontologies used by different sources, and be able to extract the semantics from each source such that they can be compatibly aligned.

The benefit of using Semantic Web technologies is that the toolsets support a number of *serialisations* (typically RDF/XML, Notation3 or N-Triples), and as such any approach that uses the Semantic Web or Linked Data supports negotiation to receive data serialised into a markup that it understands. Similarly, as mentioned above, Semantic Web technologies use URIs in order to reference and identify entities. Thus, while the above issues represent the above complexities of dealing with heterogeneous data, by using Semantic Web technologies, a degree of certainty can be achieved, regardless of the modelling and representation using by a data source. For example, RDF always defines metadata as a direcetd graph, using URIs.

In addition to modelling and representation issues in heterogeneous data use, discovery and serving of distributed data is a further challenge.

One of the ways that Semantic Web data is made available is as *Linked Data*, whereby a URI is used to identify a resource, and that URI is a resolvable http-schemed URL. By resolving that URL on the world wide web (or intranet, in the case of enterprise or private data), information about that resource, as RDF, is returned. By using such as approach for all data, the distributed nature of the web is utilised to provide a way logical way to refer to remote sources of data in the structure of the data itself. Linked Data therefore enables similar control by authors to link to offsite documents as is permitted on the world wide web. For example by using the BBC */music* URI for Chopin[1], the metadata then links to their offsite sources, including PBS, MusicBrainz, IMDB and all recordings of Chopin's music currently on the BBC iPlayer. Thus rather than attempt to provide a definitive set of metadata for every resource that an author wishes to link to, one can instead link to a definitive source. Similarly, an acaemic may wish to provide links to the papers that they have published. Rather than provide titles, ISBNs, authors, journal information, and so on, they can instead link to the URI provided by their institution's scholarly communications digital repository[2], which can then remain the centralised point of canonical information about those publications. Similarly, generic content management systems such as Drupal can also offer Linked Data through plugins (Passant and Laublet, 2008), which addresses the challenge of supporting linked data publication of existing assets.

2.2 Challenges of Combining Heterogeneous Data: Co-Reference and Alignment

In this section, we detail the related work which combines heterogeneous data, obtained from the Semantic Web, Information Retrieval, and HCI research areas. In order to address the challenge of enabling users to explore across domains, data from heterogeneous source must be combined. There are challenges that have to be addressed, in that data from heterogeneous source refers to entities using different identifiers, which is known as the co-reference problem; and work to alleviate this through alignment techniques is analysed below, in the context of previous interfaces that have combined heterogeneous sources.

2.2.1 Exploratory Interfaces of Heterogeneous Data Sources

CSAKTiveSpace (mc schraefel et al., 2004) produced an exploratory interface on to combined information about UK computer science research. The model used to query semantic stores was extended and genericised in mSpace (m. c. schraefel et al., 2003), as described in the introduction. This model allowed heterogeneous sources to be aligned to an ontology, and this mapped such that the mSpace browser can query for the union

[1]BBC /music URI for Frédéric Chopin: `http://www.bbc.co.uk/music/artists/09ff1fe8-d61c-4b98-bb82-18487c74d7b7`

[2]Support for Linked Data exists in repository software such as EPrints: `http://wiki.eprints.org/w/New_Features_Proposed_for_EPrints_3.2#Linked_Data_Support`

of the results. Some of the techniques used for Semantic Web querying build off previous work, for example in OntoBrowse (Smith, 2004) software we developed in 2004, which provided column browsing of triplestores, and CSAKTiveSpace (mc schraefel et al., 2004), which offered faceted browsing (with mapping) of UK computer science data, from a triplestore. There are other models for semantic and heterogeneous exploration, most notably "/facet" (Hildebrand et al., 2006) and "exhibit" (Huynh et al., 2007b).

"/facet" (Hildebrand et al., 2006) queries a knowledge base for ontological (RDF/S) classes, and presents these to the user, with the ontological predicates as a way to explore the space. The benefit of the /facet approach is that the only configuration required is that a ontology be formally defined. This differs from the mSpace model which was designed to work with arbitrary data, and therefore requires a definition, given the lack of an ontology. The benefit of the mSpace approach is that relationships that are multiple predicates away can be flattened into a single columnar relationship, whereas this would have to be a multiple column path in /facet.

"exhibit" (Huynh et al., 2007b) is a UI widget designed as an easy-to-deploy column browser for use in websites. It accepts data in the form of rows, with each column of data in a row being an entry in each column in the browser. The benefit of the exhibit approach is that concepts need not be defined semantically, and tabular exports are frequently available on many database systems. The downside is that the system does not scale to well past hundreds of concepts, that multi- linked columns (i.e. Composer / Composer Country) are not possible, given the flat nature of the data table form, and that provenance of data is not shown.

There is a rich source of related work in the semantic browsing space from the exploration of museums, such as (Collins et al., 2005), which matches artefacts to visitors interests, in a UK Museum scenario. One of the key motivating scenarios for the "/facet" browser (Hildebrand et al., 2006) is the browsing of museum collections in the Netherlands. There is also local work on navigating cultural heritage collections (Pitzalis et al., 2006), this time with the mSpace browser (m.c. schraefel et al., 2006) as the UI component of a richly integrated system.

(Oren et al., 2006) automatically weighs the quality of facets taken from combinations of RDF schema, and compares facet browsers like mSpace to their own browser, which allows for complex constraints to be graphically placed over arbitrary RDF data.

In the ClioPatria project (Wielemaker et al., 2008), a search and annotation framework is presented in the cultural heritage domain. This framework utilises various thesauri and semantic web techniques to enhance keyword search and clustering of results. The author notes that while they use SerQL and SPARQL as query languages, the API used to access the data goes beyond what SPARQL is specified to do, and as such, their own non-standard API is used. The use of a non-standard API is said to be a frequent source of discussion, and our work, described in this chapter, enables standardisation of the data-to-interface description. While this will not provide any standardisation to the extended querying required for some of the features of ClioPatria, such as multi-script searching, it does allow the RDF data to be described before being used in a faceted browser.

Another approach to co-reference and data alignment is the manual approach, where human operators load both datasets and make mappings for each entity themselves in a spreadsheet or similar application. People that add value to data sets are known as"Knowledge Workers" (Drucker, 1974), a term that covers any job whereby value is added to data. While, given domain knowledge, the accuracy of the information is consistent, this approach does not scale as well as automated techniques, as the output of a human is typically lower, and cost per unit is often high.

2.2.2 Mash-ups of Heterogeneous Data

Another form of data combination that has proven useful is that of "mash-ups" on the world wide web, whereby information of the same data-type are "smushed" together in a single view. One of the most common type of view is the "Google Maps[3]" mash-up, where two or more different sources of entities are combined that have associated locational geo-data, and plotted on a single map. This allows new types of analysis to become possible, with a low interrogation time. Examples of this are showing schools and houses for sale on the same map, so that parents that are looking to move home can view an area of a map and instantly see the proximity of schools to available housing. This type of view is very useful, although it is still a manual process, the sources of data must be located, and mapped to a single format, meaning that, as with the Google paradigm described above, the mash-up can generally only answer questions that have already been asked.

In order to allow users to create their own mash-ups, and therefore ask new questions, several different mash-up creation tools have been produced. The most prominent "mashup makers" are *Yahoo! Pipes*[4], *Dapper*[5] and *IBM QEDWiki*[6]. They are implemented as examples of so-called "visual programming"(Myers, 1986) tools, meaning that sources of data, and performable operations are represented on a graphical workspace as movable blocks, that can be connected together to create an executable data flow that acts to combine and *mash-up* the data sources. This does require some programming knowledge, to a level that depends on the data source; if two RSS feeds are simply to be combined, little understanding of programming would be necessary, if however, regular expressions on text is required, the barrier to entry is raised significantly.

[3]Google Maps: `http://maps.google.com`
[4]Yahoo! Pipes: `http://pipes.yahoo.com`
[5]Dapper: `http://dapper.net`
[6]IBM QEDWiki: `http://services.alphaworks.ibm.com/qedwiki/`

To get the most from data sources, it is often necessary to perform textual analysis, either for subtext extraction as in the above *regular expressions* example, or for aligning different formats, for example when telephone numbers might be in different forms. Work on Potluck (Huynh et al., 2007a) explored how casual users might want to not only graphically perform mash-ups, but also syntactically edit data fields, such as telephone numbers, when they are presented in different formats. An example is that an academic departmental web site may present telephone numbers as internal numbers without a dialling code, whereas the university web site may present full numbers. Using Potluck, the user can quickly add the dialling code to the numbers scraped from the departmental site, thereby aligning the data. This is performed using a graphical interface that lowers the barrier to entry in data alignment and mashing-up of web sites' data. This approach does not scale further than a small number of small sources, but it is very quick to achieve a result. We refer to this level of integration as **lightweight semantics**, and contrasts to the following example.

An example of data reuse and combining is given in the work by (Mannes and Golbeck, 2005), which uses Profiles in Terror(Mannes, 2004) as an example domain for exploring the use of ontologies and rules on the Semantic Web in terrorism investigations, as well as the use of trust and provenance (Golbeck and Mannes, 2006). The work documents the creation of a complex ontology, as could be used to combine data from the investigation bureaus of different countries, to generate new findings through inference. It gives the example of using inference and rules to discover unobvious connections between suspects, through third-party connections and inferred relationships. The trust and provenance work here is clearly very important, as investigators will need to provide proof to law enforcement authorities as to why people are being arrested, and why this is legally a case, aiding prosecution, in this case. This is an example of the use of heavyweight semantic technology, using OWL DL (Antoniou and van Harmelen, 2004) for formalising concepts and enabling inference, and extending the capabilities of OWL alone with rules. While this approach may be considered to be more formally complete, it is complex and expensive to produce and maintain, as shown by the ontology cost estimation models, (Bontas and Mochol, 2005), and requires all data to be matched to the ontology in order to include it. We refer to this level of formalisation as **heavyweight semantics**, opposing the **lightweight semantics** of the Potluck approach described above, which is inspired by Hendler's description that "a little semantics goes a long way" (Hendler, 2003).

2.3 Supporting Semantic Data: Lightweight and Heavyweight Approaches

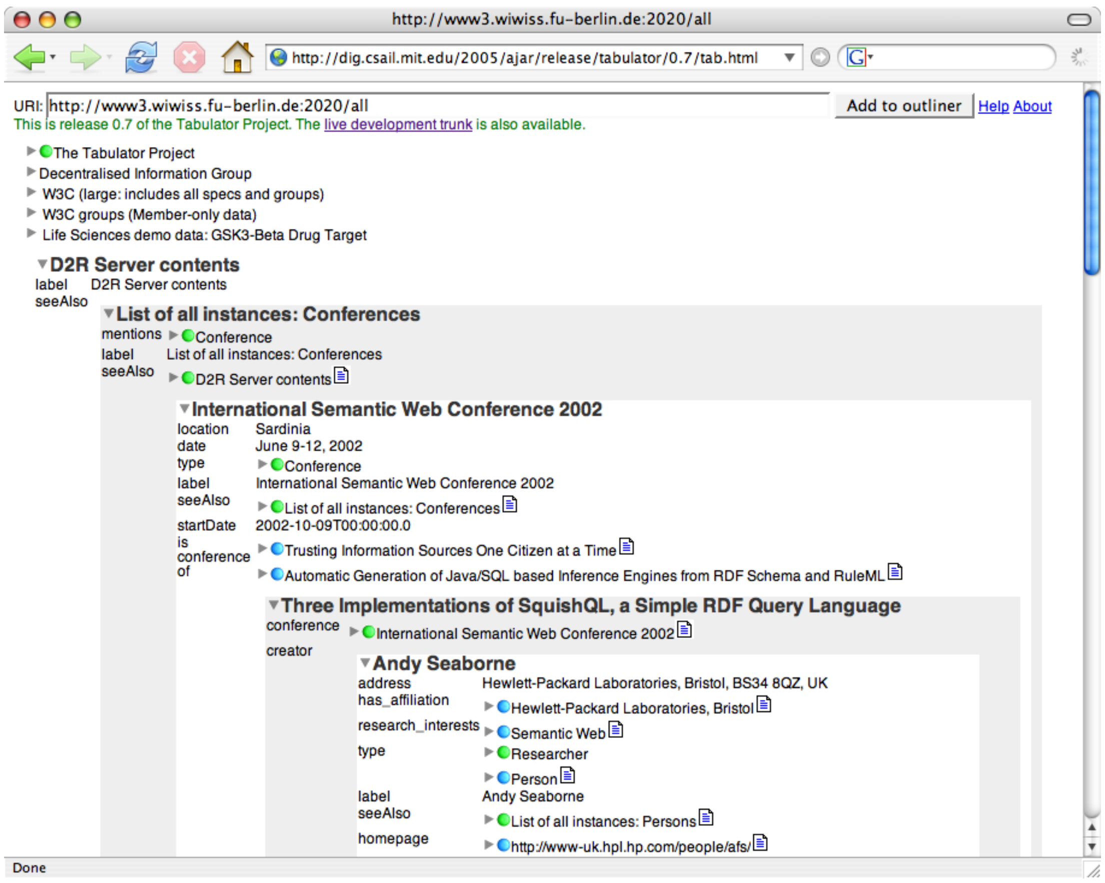
The level of semantics that our work investigates, lies in the middle ground between **lightweight** and **heavyweight**. The use of full semantics is extremely beneficial, there is however, a high cost of implementation and use, especially when one considers casual annotations by end users, and the use of semantics in aiding combining of multiple data sources. Our research in rich tags (Smith et al., 2008) has shown that it is important to minimise the amount of interaction that people have with tagging systems, keeping the interaction as lightweight and simple as possible; when the interface becomes even a little complex, its usage breaks down. This also backs up the assumption that *tagging* systems are popular due to their ease of use and simplicity. The level of semantics that I am aiming to add must therefore keep users tagging, but add value through as much semantics as still keeps users happy. An example of similar work in this area is the Haystack project (Quan et al., 2003), which developed an interface framework that allowed users to manage data such as e-mail, graphics and calendar events, adding semantics *in the background* as the user carries out their tasks. The approach is that using RDF behind the scenes enables more features to be in place, like semantic drag-and-drop, where resource URIs are traded between applications, rather than content. A driving example that Haystack uses is that of WYSIWYG HTML editors, they allow the editing of a common markup language, but without the user ever having to know about it; but if they do understand the underlying language, they are free to edit it, in an advanced interface. The use of underlying semantics, with an interface that users recognise, and is also simple-to-use, places Haystack in the same **semantic middle-ground** as our work.

For a general example of an interface to explore and combine across data sources, we can look towards The Tabulator (Berners-Lee et al., 2006), which enables the exploration of RDF data sources across the web. In RDF, entities are identified by a URI, which is resolvable as a URL on the world wide web, where there may lie an RDF document with knowledge about that entity (Quan and Karger, 2004). This means that new content and knowledge can be added to the Semantic Web, and as soon as it is linked through existing documents, can be accessed. Trend and combination analysis can be performed in Tabulator, by specifying patterns in the data, that are then automatically *tabulated* in the browser, for example, a user can load a number of RDF files and in one of them select a pattern of a class of type *geo:Place*, its *rdfs:label* and the *geo:lat* and *geo:long* of its geographical location. Tabulator will then match this pattern on all RDF files that the user has loaded, and tabulate them into a single table. The tabulated data can then be examined through viewing plugins, to interrogate the data on a map, or calendar, for example.

Compared to a manual collection of facts from web sites, having an on-map mashup, combining multiple sources, has value, as interrogation speed goes up compared to manual text comparison. This is particularly useful when one is comparing street addresses, as even if the area is known, the map will still have a lower cognitive overhead when comparing many places, compared to a list of addresses.

2.4 Problems with Existing Exploratory Interfaces

While exploratory interfaces are usefully adding value over data sets that cannot be explored, there are various problems with these systems, and challenges that existing work has attempted to address. For example, Tabulator (Berners-Lee et al., 2006), shown in Figure 2.4, is useful for visualising knowledge that has been formalised, but if that knowledge is not codified in RDF, it cannot be displayed. Even if it has been codified in RDF, there may be barriers to easy visualisation with Tabulator, for example a lot of information may be collected in a very large file; Tabulator, and, more specifically, the paradigm of client-side data analysis has scalability problems, particularly within a web browser, as is the framework that Tabulator utilises. Orthogonally, knowledge may be split among so many smaller files, that loading them all in to Tabulator's knowledge base would take the user a long time to add. Essentially, the data acquisition model that Tabulator utilises is as equally prone to the "can only answer questions that have already been asked" problem as mashups are.


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 of Trusting Information Sources One Citizen at a Time
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 2.4: Screenshot of the Tabulator tree exploration interface, showing a path opened through all Conferences, to ISWC 2002, to a paper, to the author of that paper. Each of the links fold out as a nested subtree, which can themselves be opened.

2.5 Challenges of Speed and Performance for Exploratory Interfaces

One of the challenges faced by interfaces developers (Nielsen, 1993), particularly on the web (Nielsen, 1997) is with dealing with heavily-distributed data, is optimising the flow of data to the client, so that the user is not left waiting for data that they didn't want or need to explore, which is a double-headed problem. Firstly it involves determining what information the user wants, a question of how to develop semantic data discovery systems, and how to segment knowledge into chuck of relevance for users, either before-the-fact or through query interfaces, such as SPARQL (Prud'hommeaux et al., 2005). The problem also involves how to maintain adequate performance when querying knowledge bases and engines for results, and how to push these to the user for the best possible user experience. The *de rigueur* Semantic Web tool, Tabulator has problems due to it conforming to a generic distributed data framework where the data, as RDF, can conform to any number of distributed ontologies and describe as little or as much knowledge in a file as the data provider wishes to export. Centralising the knowledge into a single store does not fully solve the problem of getting data to the UI, as the technology to query data graphs, for example in triplestores, has problems with scale when less-than-simple questions are asked.

In order to explore how to integrate data sources, we have developed the mSpace data access model, which allows arbitrary structured data to be described so that an interface can be created on top of it. In terms of data access and exploration of facetted structured data, the "/facet" browser's (Hildebrand et al., 2006) data model is similar to the mSpace model, but differs in a number of keys ways regarding the data description and user experience. Firstly, "/facet" does not require a description like the mSpace Model that we developed, instead it allows the user to explore a knowledege base, using the predicates defined in OWL ontologies (Horrocks et al., 2003), as the explorable dimensions. This is beneficial in that it does not require a description of the explorable space to be defined, outside of the ontology. The benefit of using the mSpace Model approach is that the exploration is not limited to direct predicate following, instead a list of associations can be defined, for example, the birth date of a composer, that can be explored without having to query the interim stage, in this case, the composer. Using /facet to explore heterogeneous sources would require either direct mappings of multiple sources to a single ontology, which is a heavyweight and costly task, or the invalidation of the description logic formality, as in the mSpace model, which breaks the paradigm that /facet relies on, and so would be an unsuitable solution, and as such, the /facet methodology does not support the browsing of heterogeneous sources well.

Work by (Huynh et al., 2007a) approached the problem of data integration at a personal level, by providing methods and tools to allow integration of small similar sources of data in order to complete small query tasks. One of the example uses of this work is to query over two different university contact information web sites, to populate an address book. In this example the two web sites use different patterns to describe web sites, with one using hyphens between numbers, and the other omitting externally dialable area codes. Huynh's work makes it straightforward to combine these sources, in a lightweight manner suitable for casual users, which works well for small sources. This does not scale to combining large scale or large amounts of sources, however.

As noted above, the work that exists leaves a gap in the middle-ground between the use of heavyweight formal semantics and ontologies, and the use of casual annotations, tagging and data merging in the mash-up and Potluck works. Innovation in both the front-end interface, as well as the back-end data processes was researched in order to enable the kind of *hidden semantics* that lets the user benefit from a small amount of semantics, without it hindering their use of the software, leading to its disuse.

2.6 Exploring Heterogeneous Data

In terms of information seeking and exploration activities, the more effective interface depends on the form that the source data takes. At its most basic and general form data is in the form of full text documents (usually written in a single language, but not cannot be assumed). Further structure can be present in documents, which can be used to enhance exploration of a corpus of documents, using techniques such as faceted browsing. Such techniques of exploration are discussed in this section.

2.6.1 Exploring Text Documents

A typical information seeking service such as Google will index the words present in these documents, and allow the user to query against the index in order to find matching documents. In information retrieval research, refinements to this process have been developed, on both the interpretation of the user's query, and the ranking of the resulting matched documents.

In order to better serve the needs of the user, techniques such as query expansion (Jones et al., 1972) can be used to increase the hits that a particular query will yield. Query expansion is where a base query from a user is expanded to cover additional related terms that aim to match the user's intention, without directly matching the specific terms used, so that the results are a *semantic* match, as opposed to a simple *syntactic* match. For example, a user may wish to find documents about routes of rural walks in England, and so they search for "england rural walks." A direct term match would yield results for documents that contain these three exact terms. However, this missed documents that contains the terms "english countryside walks," even though this means the same as "england rural walks." This mismatch is an example of *synonymy*, that two words have the same meaning. In a similar vein, false positive matches can be found when a term has multiple meanings and the document is not about the meaning that was intended, in an example of *polysemy.* In order to combat these problems, semantic analysis of the corpus can be performed, using techniques such as simply requerying using synonyms from a thesaurus, where “rural” is a synonym of “countryside,” resulting in a hit using this technique. Furthermore, knowledge of the stemming grammar rules of the target language (e.g. English) can help expand the query to stemmed variations of the search terms, so that “walks” and “walking” will match. More complex techniques can utilise natural language processing (Rustin, 1973) to exact semantics of the terms within documents and queries. There are also state of the art purely mathematical techniques such as Latent Semantic Indexing (Deerwester et al., 1990), which identifies patterns of word usage in documents in order to determine the weighting of matches. A mathematical approach has the benefit of being language-independent, as it does not rely on knowledge of grammatical rules of a particular language.

To help the user decide which of the matches documents are likely to be most of interest, document ranking strategies are utilised for ordering the results. Typically this is performed by assigning a weight of the closeness of the match of the document to each query term, in order to calculate an overall match weighting of the document to the whole query. The results are then ordered based on this match. In order to determine how closely the document matches the terms, compared to all of the other documents, a term-document matrix is created, using a technique such as *tf-idf*, *BM-25. tf-idf* (term frequency-inverse document frequency) is the ratio of how often a term occurs in a single document compared to its occurences in the entire corpus of searched documents. This means that given a search query such as “walks in England,” a *tf-idf* calculation shows that the term “in” is common to many documents, and thus a match to this term should have as great an influence on the ranking of the document in the results as a match to the terms “walks” and “England,” which occur more frequently.

The above techniques for information seeking work well for searching for information in textual documents. However, documents can also be annotated (either manually by people, or automatically using a processing service) with metadata, that provides information about specific facets of the documents. When such information is available, additional interface and processing techniques can be utilised to provide a richer exploratory experience. Such techniques are analysed in the next section.

2.6.2 Exploring Faceted Data

One such example of a faceted browser is mSpace (m. c. schraefel et al., 2005), a faceted browser that presents facets of a number of data sources as a columnar browser. mSpace has focused research in a number of areas. Firstly, on using the browser for exploring music, using preview cues (m. c. schraefel et al., 2003), a lightweight mechanism that allows users to preview music in zero clicks. Further to this research, a focus was placed on supporting Semantic Web data in mSpace, leading to research to show the inadequacies of scalability with alternative triplestore approaches (Smith et al., 2007), which is discussed in more detail in Section 2.7.

Faceted browsing has also been shown to be effective in the area of cultural heritage, for the browsing of image-based resources, as in Flamenco (Elliott, 2001; Hearst et al., 2002), another example of a faceted browser. The research into Flamenco showed that the choice of hierarchical categories that a faceted browser offers to the user can be more important than other aspects of the user interface.

Furthermore, there has been research into applying faceted browsing interfaces to users personal data. Exhibit (Huynh et al., 2007b) in particular, allows simple data to be extracted from local sources such as spreadsheets and published on a web site using a free lightweight widget. While this does not scale to the large collections that mSpace and Flamenco (above) are designed for, it does allow so-called "casual users" — those that do not commit to any serious training or investment to use the software — to be able to publish their small-scale personal data using a faceted browser.

One of the benefits of using small scale data is that responsiveness of performance is rarely a problem. However, we moving up to querying larger data sources, responsiveness can downgrade, and user engagement ceases. In the next section we analyse related work in the area of performance of faceted interfaces.

2.7 Challenges of Query Performance to support Exploratory Interfaces

Poor performance of an exploratory interface disengages users with their activity and prevents them from building knowledge about a domain. In this section we analyse work that attempts to address query performance that can translate into performance gains for faceted browsing interfaces.

In order to facilitate effective and efficient pushing of explorable metadata to the user, the storage and query layers that hold the combined data need to provide adequate performance to support the user experience. While developing the mSpace exploratory browser framework, and associated work, various methodologies for storing and querying metadata were explored, from various semantic triplestores, relational databases, and an in-between solution.

Since mSpace was using Semantic Web technologies such a RDF, RDFS and OWL, it also used Semantic Web technologies for querying, using a "triplestore" (Rusher, 2001) to hold the RDF, so that it could be queried using the semantic query languages

RDQL and SPARQL, as created and ratified by the W3C Data Access Working Group (DAWG[7]). RDQL was the first query language created by the DAWG and has since been overhauled and replaced by SPARQL, which has now reached the stage of a W3C recommendation[8]. Using a triplestore means that the RDF that resulting from various different data sources could be asserted into a single knowledge base in a triplestore, and the combined knowledge queried as a union set.

The triplestore software *3store* (Harris and Gibbins, 2003) was the first query solution utilised. It was one of the first implementations of RDQL that could reach any level of performance (Redland (Beckett, 2002) was the first complete implementation, but was not designed for scalability, more for completeness of meeting the specification). Subsequently, it was also the one of the first implementations of SPARQL to achieve significant scalability. Both the RDQL and SPARQL interfaces were used during my research, with SPARQL adding signficant useful features, such as keyword matching (although SPARQL in fact supports regular expression matching, meaning that this results in poor performance compared to SQL's indexable keyword matching syntax). Despite 3store providing good performance for applications such as CS AKTiveSpace (mc schraefel et al., 2004), when large datasets were created for use in the mSpace framework, the performance suffered. This is due to the high interconnectedness of the data, and that the interface makes it possible to create complex queries quickly with a small number of user clicks. The method of data storage that 3store employs also meant that the indexing available was limited, and as such, a solution was a greater level of index control was sought.

The D2R server software (Bizer and Cyganiak, 2006) was seen as a good replacement, as it supported the querying of data using SPARQL (so that the mSpace query creation would not have to be vastly changed), while also allowing indexes to be tweaked on a schematic level, as it stored the data in a MySQL (MySQL, 2004) relational database, meaning that table level optimisations such as compound indexes could be altered. Unfortunately, D2R was not the silver bullet to performance suitable for our needs, due to several problems with the way some queries were handled, as well as the large processing and memory footprint created by the use of a Java virtual machine in the implementation of D2R. Ultimately, however, one of the key problems with using D2R, or indeed, any SPARQL-based system is that there is poor support for keyword searches, due to having to use regular expressions for searches, which while rich in expressivity, is poorer than dedicated substring index matching.

In order to provide good performance for mSpace, the use of a traditional relational database was employed (in this case MySQL), so that not only was full control over queries not carried out by a third party piece of software that is not as informed of the requirements as mSpace is, but also this eliminates the overhead of an additional layer in

[7]DAWG: http://www.w3.org/2001/sw/DataAccess/
[8]http://www.w3.org/TR/rdf-sparql-query/

the query chain.

Another benefit of using MySQL is that is is extensible at a very low level. In the case of our requirements, this meant that a separate dedicated indexer could be used for keyword searching. The Sphinx [9] search indexer utilises the extensibility of MySQL allowing extremely fast keyword and subkeyword matching to be performed in-query, resulting in a scalable high-performance experience.

As performance of triplestores is of high interest to many Semantic Web-based projects, and as using the Semantic Web as the core technology for projects is popular, the benchmarking and comparison of triplestores is a research area with a lot of publications, such as (Streatfield and Glaser, 2005), (Lee, 2004). The results of the use of triplestores and query performance in mSpace in my research has also been published (Smith et al., 2007).

2.8 Challenges of Marking-up Machine Readable Data for Human Use

Once data has been opened up so that it is available for use by machines, a challenge is how to present that to users, so that it can be explored and help users to build knowledge. In order to provide value over the original human-viewable web site view of the data, generic exploratory interfaces such as Tabulator (Berners-Lee et al., 2006) and Disco address the challenge of allowing users to ask more complex questions of the data, and build knowledge using their own constraints, as opposed to fixed dimensions provided by data publishers web pages. A challenge is that while a publisher's web page is optimised for human reading, their data export contains all data they have, and thus giving it all to the user may overwhelm and not be useful for all contexts. Thus, work addresses this challenge by providing abstracted views over the data, suitable for particular contexts. In order to address that multiple contexts over multiple sources are related and would benefit the user if they could be browsed over, work has been performed in combining metadata. In combining unstructured metadata, we focus on work where pieces of data have "gaps," that are filled by using another data set, or when by adding another layer or level of data, a new facet can be created over the data. An example of this may be when a data source has location registered over cities. By careful combination of data sources, the correct countries can be derived, providing a more course grain of location that can be used to aid exploration.

[9]Sphinx: http://www.sphinxsearch.com/

2.8.1 Sources of Machine Readable Data

To address the challenge of building knowledge over multiple sources, a user or provider of interfaces has to identify sources of data that might be of interest, in this section we describe example sources, and how their export of structured data has enabled data to be browsed across domains.

There exists a growing number of providers of structured data on the world wide web. While some of these provide commercial protected data, there are a number of projects to supply structured data freely, often under a creative commons[10], such as Freebase (Bollacker et al., 2008) or other copyleft license. This not only allows it to be used for free, but for derivations of that data to be re-published.

Perhaps the most well known (and largest) of these projects is Wikipedia, which provides a number of ways to extract structured data. Firstly, there is a sizable categories hierarchy, of which articles typically belong to many categories. Additionally, there are a large number of templates that apply to different types of articles to show "infoboxes". These provide a restricted set of key/value pairs (including in-site links) that can be mined semantically, for example a musical band infobox describes members, type of music, and links to the discography. Figure 2.5 shows a part of the infobox used on the page for the band "The Happy Mondays." It shows the Town and Country of origin, Genres, Years Active, Label, Associated Acts and Members.

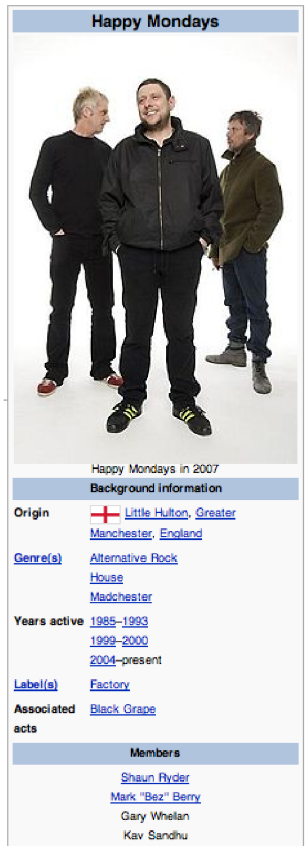
The dbpedia project (Auer et al., 2007) aggregates the metadata present as wikipedia infoboxes, allowing them to be queried using the Semantic Web's standard query language SPARQL (Prud'hommeaux et al., 2005). They provide a SPARQL endpoint on the web, so that Semantic Web applications can utilise Wikipedia data as they would any other Semantic source.

Another project, FreeBase (Bollacker et al., 2007), has a similar ethos of creative commons freedom and universal editability to Wikipedia, however it takes structured data very seriously. It has introduced the concept of Types and Types Editors. A type defines a class than a page can be a member of, and a set of fields that a type can have. For example, something of the type "Person" is defined as having a Name, Gender, Place of Birth and so on. A Type Editor is the person, or persons that have been assigned the credentials to edit the fields that a type defines.

These projects all utilise the Creative Commons license on their data, with strict removal of non-compliant data, and as such are legally safe to repurpose in other projects. It is this policy and community that enables the levels of growth and participation in community-editable data that has been witnessed recently.

In order to enable structured data sources to be able to be browsed and to allow knowledge

[10]Creative Commons: http://creativecommons.org/



Happy Mondays

Happy Mondays in 2007

Background information	
Origin	Little Hulton, Greater Manchester, England
Genre(s)	Alternative Rock House Madchester
Years active	1985–1993 1999–2000 2004–present
Label(s)	Factory
Associated acts	Black Grape

Members

Shaun Ryder
Mark "Bez" Berry
Gary Whelan
Kav Sandhu

FIGURE 2.5: Wikipedia Infobox for The Happy Mondays, taken from Template "musical artist 2"

to be build across the sources, technologies that address the challenge of combining and aligning those sources can be used. In the next section, we analyse the related work in this area.

2.8.2 Technologies for Data Combining and Alignment

There is an assortment of technologies available and in-development for the publication, definition and sharing of structured data online. These vary in structure, strictness of semantics, and reference-ability to and from other data sources. In this section, we analyse work that address the challenge of enabling users to browse across sources, through dynamic combining of data.

Starting from the wilder side of the relevant technologies, there is the freeform collaborative annotation phenomenon known as "tagging." This allows users to annotate a resource with a string (Voss, 2007). While implementations of this idea often vary slighty, there are two key issues with tagging. That of synonymy and polysemy. Words are synonymous if they can mean the same thing, that is to say that two documents tagged with "notebook" and "laptop" respectively are about the same subject, however due to different words being used, they may not be associated together. Polysemy is the problem that a single term has multiple meanings, leading to the issue in tagging that two documents become related incorrectly, such as a document about sausages in buns, and a document about warm canines.

On the other end of the structured data spectrum sits the work on the Semantic Web (Berners-Lee et al., 2001). This programme, led by the World Wide Web Consortium (W3C) is primarily concerned with the encoding and interchange of semantically rich metadata. Thus far, in the structured data work, the output of this effort has been the formalisation of the Resource Description Framework (RDF) (rdf, 2004) and the OWL Web Ontology Language (OWL) (Horrocks et al., 2003). These mark-up standards allow the description of referenceable-over-the-web metadata, utilising Uniform Resource Identifiers (URIs) (Berners-Lee et al., 1998) so that multiple documents can refer to identical resolvable instances. The instances are resolvable in that they can be accessed directly by their URI (using the URI as a URL), to retrieve an OWL/RDF document about that instance.

One might wonder, given the richness of structure and semantics offered by OWL/RDF, why tagging has become so popular, while uptake of use of Semantic Web technologies have remained low. The key difference here between tagging and full-on Semantic Web is that when tagging, the user does not need to find and refer to ontologies, or worry about the semantics of their annotation. For example, they can tag a photo with "tiger," meaning that it has a photo of their pet cat, named Tiger, somewhere in the photo. To do the same in a Semantic Web annotation, the instantiation of a cat, Tiger in this case, would need to be defined, against a relevant animal genus ontology, and the specific semantic of the annotation found, or defined in an ontology, i.e. that the cat is the subject of the photograph. Without the semantics, the tag is ambiguous, is this a photo of Tiger? a tiger? taken with a piece of equipment called tiger? at a place called tiger? by a person called tiger? about a tiger? The list is endless, and this complexity is entirely bypassed when a flat tagging system is used.

As an interim, or bridging system to bring semantics to the masses, there has been work in using shared vocabularies within tagging systems, as a lightweight way to disambiguate, and add some lazy semantics at the time of tagging, often using an autocompleting tag box, to simpify. Wikipedia's[11] entries are frequently disambiguated, and there is work to use these definitions to disambiguate tags (Cucerzan, 2007), and tag clustering techniques as in (Yeung et al., 2007), as a post-hoc approach to shared vocabulary disambiguation, which successfully utilises context and tag co-occurrence to determine meaning. For example, autoamated analysis to determine when *sf* means San Francisco and when it means Science Fiction, based on co-occurrence with terms such as California and Star Trek.

In order to address the challenge of providing users with the ability to specify such

[11]Wikipedia http://en.wikipedia.org/

meaning themselves, other systems have tried to extend flat tagging. In the next section, we analyse work that addresses the challenge of specifying semantics through tagging.

2.8.3 Extending Flat Tagging

When allowing users to tag items, the meaning of the tags can be ambiguous. One technique that addresses this challenge is to extending tags to being non-flat so that the semantics are minted at the time of tagging. There are various ways that extensions to flat tagging (as described above) have been researched, to add levels of hierarchy and categorisation, as well as adding semantics to inform and disambiguate the meaning and content of tags, these are detailed in this section.

One of the methods of providing disambiguation of a tag is to share vocabularies among users and systems. While having shared vocabularies means that a collection of tags is disambiguated, there is no guarantee that the disambiguation terms used have been shared outside of a local collection, i.e. when combining or cross-referencing documents across the internet. In this instance, it is necessary to refer to an external definition document: the ontology. This is where the "big iron" semantics come in to play, and when the complexity of the operations increases. There is much work on Ontology Mapping, so that two documents that refer to identical semantic concepts, but to different ontologies can interchange, by mapping the ontologies to each other. There are different way to perform this: though cross-ontology sub-classing, cross-ontology equating, and the use of "upper merged ontologies."

Upper Merged Ontologies, such as SUMO (Niles and Pease, 2001), are ontologies that define thousands of "common terms," usually taken from a wordlist (such as WordNet (Miller, 1995)) or a dictionary. Ontology authors, or third-parties, are then encouraged to sub-class their ontological classes from this upper ontology, such that when enough ontologies do this, inference between ontologies can perform ontology mapping. For example, I may define an ontology that includes a class of "Person." I then sub-class this form an upper merged ontology's class of "Person," and so does the author of another ontology. If OWL inference is then performed over documents that utilise either ontology, the classes are equated.

Another approach of publishing general structured data online is to embed structured data in to existing markup structures. Known collectively as "microformats," there are different methods one can use to embed. One of the most common examples, called hCard[12] of microformat implementation is to use specific CSS classes in HTML in order to identify that some blocks of text on the page represent parts of a VCard (Dawson and Howes, 1998a), for example, the following encodes the "Country" element:

[12]http://microformats.org/wiki/hcard

```
<div class="country-name">USA</div>
```

Further to adding semantics to tags and documents, a challenge exists when combining more structured hierarchical metadata sources. In the next section, we analyse work that addresses the problems faced when combining hierarchies, in order to allow users to explore the combined set.

2.8.4 Considerations when Combining Hierarchies for Browsing

When combining hierarchical metadata structures, in particular for allowing users to explore them, there are numerous challenges in ensuring that both the semantic structure of the hierarchies are not compromised, and that the users are not overwhelmed with deep and broad hierarchies. In this section, we analyse work that addresses this challenge.

Hierarchical categorisation is commonplace amongst datasets, from the venerable Dewey decimal library classification (Dewey, 1951) and the Library of Congress Subject Headers (Mischo, 1982), to product categories used in online shopping web sites. The use of a hierarchy breaks up a large corpus of subjective terms into a subsumptive tree, designed to prevent a browsing user from beign overwhelmed by choice, during an information finding task.

When combining two data sets that use different hierarchies, there are some considerations to be made. A naïve approach is to link hierarchies where terms intersect. Co-reference issues aside, this approach suffers as it can create very deep hierarchies, which have been sown to be confusing to users that are without domain knowledge (Hearst, 2006). Thus, there is work into the creation of automatic shallow hierarchies for faceted browsing, such as Castanet (Stoica et al., 2007), which has demonstrated a technique for squashing deep hierarchies automatically.

One of the most common methodologies for electronically storing and archiving information is through digital repositories. Like their physical counterparts, these repositories are managed by editors and/or librarians that are typically categorised with structured, often hierarchical, metadata. Work into how to share and expose these metadata structures faces the challenge of how to engage users to explore these structures. In the next section, we analyse work that addresses the challenge of exposing and combining structured metadata from digital repositories.

2.8.5 Structured Data and Digital Repositories

In order to allow users to explore and find information in digital repositories, structured and hierarchical metadata is used. One of the challenges in this area is how to allow users to browse structured repository metadata across archives. In this section, we analyse work that addresses this challenge. One of the leading supporters of structured data interchange are online digital repositories, under the standardisation efforts of the Open Archives Initiative (OAI) (Nelson et al., 2002). Digital repository software such as EPrints (Gutteridge, 2002), DSpace (Tansley et al., 2003) and Fedora (Payette and Lagoze, 1998). The support for data interchange began when Dublin Core (Weibel et al., 1998) was ratified in 1995. This described a set of specifically named elements under the Dublin Core namespace, that can be used to mark up documents and works that have, for example, titles, authors, dates of publication, and so on.

The OAI introduced a protocol for metadata interchange in 2001, known as the OAI Protocol for Metadata Harvesting (OAI-PMH) (Van de Sompel et al., 2004), which enables the export of Dublin Core modelled XML metadata. OAI are currently in the process of developing an Object, Reuse and Export (OAI-ORE) (Lynch et al., 2007) protocol for the interchange of digital object metadata, which is being designed to be abstract enough that it can be used for any type of respository. It is not yet clear how this will interface Semantic Web research.

The above technologies provide different ways to enable transmission of (structured) data representation across a distributed network such as the Semantic web. These vary from tags, which are structurally and semantically light, through to RDF/OWL, which enforces a high level of both structure (albeit optionally XML or N3), and differing high levels of OWL description logic to drive the semantics.

Open Guides, for example, has opted to enable metadata output through RDF, designed for mashup use, and data reuse. This is also an open system, meaning that content providers can opt to download and alter the software, changing the supported metadata, as well as how this is output in export. This leads to the benefits of closed systems like Freebase, in that free (creative commons licensed) data is available in a cleanly exported way (RDF versus API), and avoids the pitfalls of a closed non-distributed system, which Freebase ultimately is.

Designing a system for use in mashups is something that has been a large factor in the success of photo-sharing service “flickr”[13], (as well as being cheap, fast and simple-to-use). The ability to write tools has let to a community value-add to the service[14] , and one to which would have been technically possible without an API, would not have sparked the same scale of development. Thus, despite the un-semantic structure of the flickr data, there is enough value in the data that many tools have been created and used, which shows good promise for further development in this space.

It is unclear as to whether the future uptake lies in distribution through queryable interfaces, such as flickr’s API, and SPARQL open knowledge endpoints, or in the distribution of small files of data, such as RSS feeds, last.fm’s XML outputs and RDF

[13]http://flickr.com/
[14]http://bighugelabs.com/flickr/

FOAF (Brickley and Miller, 2005) files. The distinction is slightly less important when dealing with the Semantic Web, since the design of SPARQL has been forward-thinking enough that the results of a query (with the query embedded within the URL) can be returned as RDF, meaning that if one supports an SPARQL endpoint, they get the RDF file support gratis.

Further to exposing information to the Semantic Web, a challenge is in how to expose it to users, and how to deal with the various formats and serialisations that the Semantic Web supports, and how to deliver data to the user in a timely manner, using these technologies. In the next section, we analyse research into the Semantic Web that addresses this challenge.

2.8.6 Semantic Web Data Transformation Tools and Techniques

The Semantic Web delivers machine readable marked-up knowledge over the web, however a challenge of how to leverage it to enable users to explore and build knowledge over the data from the Semantic Web. In this section, we analyse work that addresses this challenge.

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[15] enables linked data to be exposed from a SPARQL end-point and D2R Server Bizer and Cyganiak (2006) 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.

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

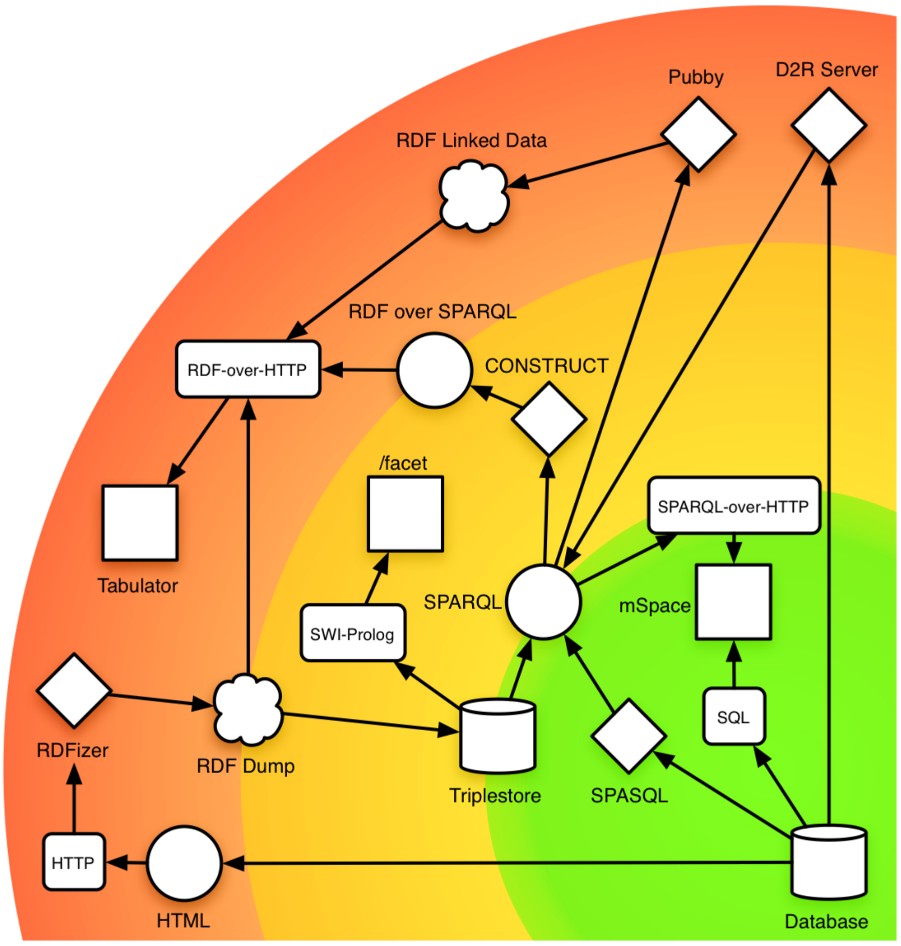


FIGURE 2.6: Interactive performance of Semantic Web data access and knowledge transformation technologies. Technologies in the green layer perform the fastest, while those in the red layer have the worst performance. Technologies on the edges of a layer being to either layer's performance, depending on use.

In Figure 2.6 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.

Additional techniques of data transformation and creation can also be utilised, such as the use of Rule Intechange Format (RIF) rules (Wagner et al., 2006), which denote how to transform data, based on whether that data conforms to rules, for example converting temperature readings in Fahrenheit to Celcius.

Additionally data publishers need to be able to determine which ontologies best fit their data, and research such as AKTiveRank (Alani et al., 2006) presents a technique for ranking ontologies, according to the coverage of an area, compared to the needs of a publisher.

Once data is publised on the Semantic Web, discovery tools are required in order to allow users and publisers to find data that conforms to specific patterns, or uses particular ontologies. The semantic search engine Sindice (Tummarello et al., 2007), offers such as service, offering lookup by URI as well as IFP. As such, users can state that they wish to find any information about a specific URI, or by the value of an property, such as any resource with a *label* of “University of Southampton.”

The heterogeneous data that the Semantic Web now provides is challenging to browse across. User interfaces have addressed this challenge with a number of techniques to allow users to move through heterogeneous sources, which we analyse in the next section.

2.9 Pivoting across Heterogeneous Data Sources

In this section we describe the challenges when dealing with distributed and arbitrary data, how work has addressed the challenge using a triplestore approach for faceted browsing, and other related work in the challenge area.

2.9.1 Dealing with Distributed and Arbitrary Data

The nature of the Semantic Web is that it is distributed, which presents challenges in automated gathering of data across the web. In this section, we analyse work that addressed that challenge.

The Semantic Web’s distributed data model means that a services providing data to an interface must determine how much of the available data to gather, when to gather it, and have an approach to deal with updates. It is only then that the interfaces can decide what of the data to display, how to display it, and how the user is able to interact with it. Additionally, the extensibility of ontologies means that a data source may contain unexpected additional data, such as extensions to a FOAF file, and hence a browser requires an approach to deal with that data: whether to display it, filter it or ignore it entirely.

Deciding on how to utilise data that uses arbitrary fields is not simple, and there have been a number of approach to it. Fresnel (Pietriga et al., 2006) is a vocabulary that allows fields to be marked up with HTML elements - these definitions are known as Fresnel lenses - so that when Fresnel is run over a piece of RDF, known elements are rendered using HTML, for use in a web browser. This approach works well, because it is extensible - anyone can mark up any data, and publish their lens, then by combining lenses that refer to the fields your data uses, you can produce a visual representation of your domain. Fresnel's strength is in its simplicity, it embraces the Semantic Web's ethos of publishing a small amount of data to achieve a task, so that people create decentralised definitions of how to deal with data, that can be used as appropriate. The approach works well for rendering data, and as such we have designed our approach to Facet Ontology in a similar way, with the difference being that Facet Ontology definitions describe how metadata connects together, so that an interface that allows a user to query data and constrain results can be produced.

Another approach to the problem of dealing with arbitrary fields is to show everything to the user, and allow them to define a pattern to match a subset of that data. Tabulator (Berners-Lee et al., 2006) takes such an approach, with a particular focus on supporting direct retrieval of RDF documents. Specifically, tabulator demonstrates the ability to resolve the identifier of an entity (its URI) as a URL of a retrievable RDF document, which tabulator then allows the user to explore. This interaction is supported continuously, so that a user can follow the graph of assertions from one RDF to another, to another and so on. Tabulator can do this without knowing anything about the RDF, since it render the metadata using only its structure - specifically by transforming the graph of the metadata into a collapsed hierarchical tree that the user can expand as necessary. Tabulator's approach to pulling in extra data is to allow users to specify a pattern to match over a hierarchy, by selecting fields of interest. Tabulator then converts this visual hierarchical pattern into a SPARQL query that is execute over the union of all RDF documents retrieved in the current browsing session. This works for the small-scale, since it is a powerful way to pick any attributes from a document and display them, without the need to write a SPARQL query manually. Scaling up with this approach is harder, because the tree of relationships becomes very large, and is not possible to view all at once. Additionally, tabulator's client-side approach means that all data is processed and aggregated on-demand locally in a web browser, which is very resource intensive, resulting in a low limit on the size of data that is supported. In order to support larger amounts of data, a back-end approach of using triplestores to store large amounts of RDF is typically employed (Broekstra et al., 2002).

2.9.2 Using a Triplestore Approach For Faceted Browsing RDF

By using a triplestore, an interface does not need to directly deal with retrieval of RDF documents from the web; that task is offloaded to the process that populates the triplestore, allowing a static set of data to sit indexed on a server, ready to be queryied efficiently. Traditionally using a triplestore would mean foregoing the gathering of additional linked data, however there is ongoing work by triplestore vendors to intelligently gather linked data, based on queries used, and as such we suspect that parts of this challenge may be solved server-side. Thus, using a triplestore for data storage means that a larger amount of data can be used, shown and interacted with, the challenge of how to display arbitrary data remains unchanged, as does the problem of how to interact with it.

One approach to providing a faceted browser with data from the Semantic Web using a triplestore was utilised in mSpace (m. c. schraefel et al., 2005). mSpace supported the querying of stores using RDQL (and later, it's successor SPARQL), meaning that it does not suffer from requiring a user's computer to load data on-demand, allowing the interface to scale up further than Tabulator. However, mSpace requires a definition, known as the "mSpace Model," which specifies how data connected, and how to generate queries to retrieve data from the store. While this means that the mSpace browser does not have deal with extraneous RDF triples itself, it requires a description of the data to be made ahead of time. This trade-off means that the browser itself can only ever deal with data that had been marked up, and that any external integration, for example across interfaces, cannot be achieved. mSpace achieves a scalable approach to querying faceted data, by utilising semantic web technologies as a bridging mechanism between heterogeneous data sources and a single interface. It does this by being optimised for rapid real-time complex queries - something that is not possible if the RDF data is gathered on-demand.

Similarly, /facet (Hildebrand et al., 2006) presents a faceted browser that uses a semantic store as both its schematic definition and set of records, meaning that the structure of the data is used as the model. This works by querying the store for all classes, and presenting them to the user as possible starting facets. Following making this choice, all predicates that connect to this class are then available as facet columns. This approach is beneficial in that the use of RDF means that in order to add additional facets, additional triples can simply be added to the store that utilise a new predicate. Since the configuration of the interface is performed dynamically based on the data, an update of the store's contents is reflected in the browser automatically. When the RDF used is carefully structured, and filtered for use in /facet, the results work very well, providing an intuitive interface for users, as well as an easy-to-understand and configure system for administrators. In the case where data is not so consistent, for example when RDF data is pulled from the wild Semantic Web, the methodology is unsuitable. This is because the Semantic Web is a general-case open world system, where an RDF file can contain information on anything at all, not just a single domain, and not just information that was requested. Since /facet has no definition model, and relies entirely on the structure of the data to provide the configuration for its facets, it would create a large number of sparsely populated facets that would be of limited use, and this problem would grow as the amount of "wild data" was loaded into the browser.

2.9.3 Polyarchies to Support Exploration

Existing approaches to browsing Semantic Web data have been limited by the amount of data they can scalably use, and the need to have detailed ahead-of-time mark-up of data, in order to determine what data to show. A possible solution to the problem of having a better trade-off between ahead-of-time mark-up of data, and scalable interfaces for arbitrary queries is through the notion of a pivot. This notion is similar to the "visual pivot" notion explored in (Robertson et al., 2002b,a), where hierarchies are extracted from different databases, and pivot points were designated, where point hierarchies intersected at common instances. For example, consider an organisation that holds a number of management databases, where a person can be contained in two different databases, a pivot point exists for each person that is in both databases. A 3-dimensional visualisation is available to visually animate the transition from one hierarchy to another around the pivot point, to show the user how the different hierarchies related, and to be able to move from one related table to another to explore relationships across data. These intersecting hierarchies are known as "polyarchies," and have been formally compared to the representation model of facet as used in mSpace, and the structure used in ZigZag (McGuffin and m. c. schraefel, 2004), where a populated taxonomy was presented to enable comparison of the differences in representation.

2.10 Summary

In this chapter we have presented the state of the art research into the publication, exploration, combination, alignment and querying of heterogeneous web-scale data.

The related work has addressed a number of key issues. It is now possible to publish data online using the Semantic Web and Linked Data, although challenges remain in how to automatically align information from multiple sources, and how to disambiguate co-references of named entities in order to make it possible to utilise data from multiple sources in rich ways. Work in faceted browsing has brought lightweight facet interfaces to publishers of small scale data, so that casual users can incorporate powerful interfaces onto their web pages. Challenges remain in how to enable users to support large scale data in the same way, and how to deliver performance over large scale data with faceted interfaces. Work on how to abstract data sources has addressed how to renders patterns of Semantic Web data, and a challenge remains in how to apply this to faceted browsers, in addition to static rendering of sources. Preliminary work into pivoted browsing of data has demonstrated that cross-domain browsing is useful, and a challenge has arisen in how to apply cross-domain browsing in the general case, across heterogenous data sources.

Chapter 3

Requirements for a System to Enable Users to Explore Heterogeneous Data

In order to scope our work, we must first know the motivating requirements that will target our research. Therefore, in this chapter we will outline the requirements based on the following three problems:

1. How to enable people to explore unfamiliar domains
2. How to give users rich features for browsing without sacrificing the performance that makes them want to use it
3. How to enable users to make connections across data sources

We begin in the next section by outlining the requirements to enable people to explore unfamiliar domains.

3.1 Problem 1: How to enable people to explore unfamiliar domains

The first problem is how to present information to users when they have limited knowledge of a domain. Thus, care must be taken to allow the user to explore the domain, its terminology and how information relates to each other. We know that an exploratory interface can enable users to explore unknown spaces (see Chapter 2), and thus, given information about a domain, we can construct an exploratory interface to aid the user. Thus, we begin by detailing the problems with describing datasets so that they can be explored in an interface.

3.1.1 The Challenge of Describing Datasets For Exploratory Interfaces

In order to achieve an abstraction over datasets and facilitate the creation of exploratory interfaces, we outline the following challenges:

1. How to enable users to define an abstracted view over a structured data source, using an intuitive interface that can read Linked Data in a similar way to Tabulator (Berners-Lee et al., 2006), by saving that exploration as an abstracted view over the data that can be used and extended by others.
2. How to lower the barrier to entry of creating a faceted browser over that abstraction, so that the user can explore the data right away, and share that experience with other users.

One of our goals is to enable users to be able to explore information from the world wide web, and thus our requirements reflect that need. Specifically, we have seen an increase in the release and publication of factual data sources from governments, which people want to explore and analyse. Specifically, initiatives such as the UK Government Open Data [1] and similar "open government" initiatives in countries such as Germany, USA and New Zealand have seen the release of data sets that cover aspects of government activities such as healthcare, policing and housing. These data sets can aid in answering questions users have, and while in some cases the use of a single source of information will yield the answer that a user needs, more complex questions require the use of information from multiple sources. Hence, the combination of two or more data sets can provide an entirely different service than the original data sets were intended.

Similarly, consider the following scenario where a user wishes to use information online to help with their decision making:

> *A user is planning a business meeting in the city, and knows which tube stations are convenient for her contacts, however her knowledge of those areas is not recent, and she wants to confirm that the restaurants she is considering are still operating, and still of a high enough quality for a business lunch. Additionally she wishes there to be a branch of her usual coffee shop nearby that she can work in using the wifi internet that she subscribes to. Information about the restaurants can found from a number of review sites, and the information on coffee shop branches can be found on the coffee chain's website, however there isn't a site that combines this information.*

Given this example, it is therefore desirable for data sets to be repurposed for a multitude of tasks, this is beneficial because it enables users to use emergent behaviour where the data sources can be combined to fulfil unforeseen tasks.

[1] HM Government Data: `http://www.data.gov.uk/`

Where research questions can be anticipated, and are required by multiple people or institutions, or are simply of special interest to technically-adept specialists, mash-up services that integrate disparate data sources are created. Such services perform the integration of heterogeneous sources, and often provide an exploratory interface over the aggregated data, in a way that allows users to answer questions that are relevant to the domain. However, as questions get more complex, or the use of multiple services is required, the existence of an integrated data service over specific data sets cannot be assumed or relied upon, and a methodology that enables the integration of data for a specific task is limited, compared to a methodology that enables users to integrate the data sources of their choice so that they can extract and explore data themselves to suit their task. In such cases, users must either perform collation of results from separate queries over each data source individually themselves, or, if they are technically able, create an integrated service themselves. The process by which a user would go through to make an integrated service themselves has become more feasible over time, because tools to enable user-created mash-ups have been released.

Thus, we require the ability for users to be able to describe datasets, so that they can create integrated services over arbitrary datasets, without the need for technical knowledge about data extraction techniques.

In the next, section we discuss the related work also contained in our literature review but has contextual importance to the area of user-created interfaces over heterogeneous sources.

3.1.2 User Created Faceted Interfaces

Users are able to publish material, including data, on the World Wide Web, and there is a growing concern of how to access that material. Discovery of data can facilitated by the use of search engines: users can submit their pages to search engines, and optimise their content in order to ensure that their context contains potentially popular search terms. Locating a particular site is not the end of a user's interaction, the user interacts with the site to find the information that they seek, and there are a number of tools available to aid a publisher in making their data more explorable and usable. Such as simple HTML and JavaScript add-ons that one can add to an existing site to make the interaction more user friendly. If a publisher already has a set of data in a table, that table can be upgraded so that it can be sorted by any of the columns, with the addition of a sortable table called *sorttable*[2]. This gives the benefit to the user that they can view the tabulated data indexed by any of the fields, and thus tailoring the data to their needs. This approach is limited, as it does not allow the user to perform any filtering of the data, and thus the data is restricted to small data sets where the user is interested in the full data set, and not any smaller subset of the whole data set. One way to filter a

[2]sorttable: `http://www.kryogenix.org/code/browser/sorttable/`

tabulated data set is through faceted browsing (as described previously in Section 2.6.2). One of the ways that users can enable faceted browsing on their own web-published datasets is by using the Exhibit (Huynh et al., 2007b) JavaScript widget. Similar in its approach to *sorttable*, above, Exhibit extracts each column in a tabulated data set as a facet, allowing the dataset to be filtered by any of the attributes in any of the columns of the tabulated data. By using this faceted browsing approach, Exhibit allows larger data sets to be browsed in-situ on a webpage.

The approach that Exhibit takes is to keep its logic and functionality on the client, to benefit from an easy installation for the data publisher, with almost no configuration required. Specifically, no configuration is required because Exhibit exposes each available data field as a facet, instead of having to separately configure which data to use in the interface. The task of customising the interface is therefore instead abstracted to editing the raw data to remove fields or values that the publisher does not want to appear in the interface. In order to aid this task, a tool such as Potluck (Huynh et al., 2007a) can be used to apply edits to multiple fields at once.

However, Exhibit's approach of processing the data on the client-side begins to suffer from performance problems once the number of records exceeds a few hundred records (depending on client computer specification). This is because the amount of resources, such as memory and network bandwidth, are limited for web browsers. A server-based approach can be taken instead, to overcome the limitations of client-side processing.

Thus, we require the ability for users to create scalable exploratory interfaces, without technical knowledge of interface design, development or performance engineering.

3.1.3 Browsing Linked Data

To browse pages on the the World Wide Web, a web browser is used, which allows HTML pages to be shown to the user, with embedded hyperlinks that can be clicked to lead to other pages.

Creating a browser for HTML documents is relatively straightforward: render pages and support links. We acknowledge that this description is oversimplified, but is valid because there is little question about what the browser is supposed to represent: documents. We know what documents look like and how they behave; we have hundreds of years of practice with many kinds (m.c. schraefel, 2007). This model is undermined, however, as soon as we talk about Linked Data, rather than documents. What does it look like? How do we read it? How do we present it in order to make sense of it?

A few approaches have been proposed: represent the data as the big graph of connections it is. While this representation may be valid, it has limited utility if one is less interested in what node is next to or far from another node (Karger and mc schraefel, 2006) and more interested in understanding what happens when that information is blended. If one actually wants to query the data in the graph, therefore, there is currently a manual approach, Tabulator (Berners-Lee et al., 2006), that lets a person identify and pull in various linkable RDF sources. This User Interface (UI) enables one to inspect the relationships of the sources, select properties of interest and then run a query to "tabulate" a result set. This approach has the advantage of allowing arbitrary queries to be constructed across arbitrary RDF resources, but costs time to find, load and explore those attributes and then render the query. Currently, the size of RDF data sources that can be effectively rendered is small. Despite the limitations, the principles in the UI express a particular approach to exploring RDF data: in a first generation Web way (pre search engines) its design implicitly assumes the user must find the resources to feed the Tabulator themselves, explore each resource for attributes of interest and then define the specific query.

Another less manual approach is to pre-present data sources related to a topic so that the data in those sources can be persistently explored from a variety of contexts. This approach is generally referred to as faceted browsing. In these interfaces, attributes of the data space are presented as categories, and instances of those categories are made visible for selection. Generally, selection of an instance in a facet acts as a filter on what instances appear in other facets. This model facilitates representing and exploring relationships in linked data in particular.

There are many approaches to presenting and interacting with facets as demonstrated in exhibit (Huynh et al., 2007b), /facet (Hildebrand et al., 2006) and mSpace (m. c. schraefel et al., 2005). The reason perhaps for the variety of faceted approaches in semantic web UIs (and their increasing occurrence in online shopping sites) is that often classes or predicates of the RDF sources can be readily represented as facets that have some kind of hierarchical relationship to other attributes in the data, making it easy to expose and so explore those inter-relations. Thus, rather than having to take the time to collect all the sources manually (though one can imagine a search engine designed to gather sources about a topic to feed these UIs), a site exists that has already gathered (and usually cleaned the data sources) to be integrated by a particular UI service. What faceted browsers have so far required however is that someone define a particular domain for the browser, and that these collections of integrated sources persist beyond the life of any individual query.

Thus, in the recent history of the semantic web, the user interaction tradeoffs have been either to be able to pull together any RDF source or set of resources (providing they aren't too big) in a viewer like Tabulator dynamically to run some queries on them via direct manipulation, and have a very raw, in the wild user experience i.e. requiring technical expertise, or users have been able to explore and query the pre-defined, integrated data resources using rich interaction tools, but representing only a few domains. Great if that's a domain of interest; not so great if one wants that UI power over a different set of resources. One might say well go ahead and pull together those sources using the tools available to create them. Despite the increasing facility of such tools to sling these UIs over data sets, the cost of effectively having to create a persistent data space if one just wants to run a single query and move on, is obviously too high.

Our goal has been to look at ways to bridge the gap between refined, effective, if tame Semantic Web oriented UIs like exhibit, /facet and mSpace and the in-the-wild arbitrary query UIs like Tabulator, that are neither currently as scalable nor have tested as well in terms of general usability with neophyte users in particular as the tame UIs. In other words, we want to blend the advantages of on-demand, one-off queries with the interaction facilities of these more archival data domains. One approach has been to start with a predefined domain and consider how to add in new information on-demand from outside that representation to facilitate new queries. That is, a person exploring a domain that has brought together a set of data resources will be able to see other resources that may be relevant to attributes in that domain and include them in the space if they wish. So someone looking at the term Baroque in a music domain would see that there is information about Baroque as architecture also available. They would then be able to pivot over to look at that data in its own context, or sweep in that data into their current context for direct exploration/comparison.

While not yet completely facilitating arbitrary queries across arbitrary sources, this approach provides two attributes we need to get to that more complete query in-the-wild solution. First, it lets us look at how we can enhance the value of well-tested usability techniques with wild or semi-wild data sources dynamically, and second, will let us move towards better understanding from how these interactions work of how totally wild/arbitrary linked data queries may be handled in an effective and scalable UI. We hypothesise that even in the fully wild case, the principles of what we are proposing to extend in these pre-defined domains will also benefit fully arbitrary queries across dynamically linked data for dynamic queries.

3.1.4 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[3] 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 (Berners-Lee et al., 2006), 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

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

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 semantically-specific rendering is performed, such as was explored in the Fresnel project (Pietriga et al., 2006). *Fresnel* defines the concepts of *lenses*, which are human created descriptions 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 (Brickley and Miller, 2005) or vCard (Dawson and Howes, 1998b) 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 (Smith, 2004) software we developed in 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 (Hearst, 2006). There is work into the creation of automatic shallow hierarchies for faceted browsing, such as Castanet (Stoica et al., 2007), 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 (Horrocks et al., 2003), possibly combined with RIF (Wagner et al., 2006) 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.

Thus, our requirement is that abstractions can be defined over data sources, and that those definitions describe the facets and attributes of the data source. Faceted browsers must be able to use the definitions, presenting the data in an interface that users can use to explore the space.

3.1.5 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 displaying 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 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 (Hyvonen et al.; Makela et al., 2006) 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 (Suominen et al., 2007) 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.

Thus we require a scalable browsing experience, where 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 conclusion, we have determined 5 key requirements for the problem of how to enable people to explore unfamiliar domains:

1. The ability for users to describe datasets.
2. An interface for users to create faceted browsers from descriptions.
3. Methods for ordinary users to browse linked data.
4. Creation of formal abstractions of datasets.
5. Scalable creation of faceted abstractions.

In the next section we enumerate the requirements for how to provide richer browsing functionality while maintaining high levels of performance.

3.2 Problem 2: How to give users rich features for browsing without sacrificing the performance that makes them want to use it

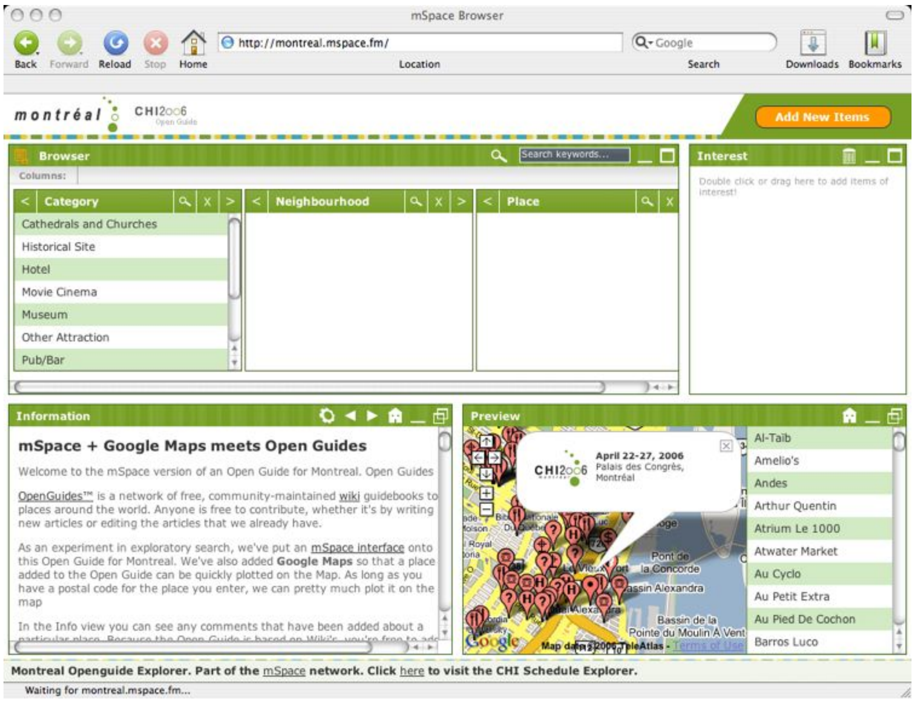
In this section we describe additional features that we require in order to solve challenges in faceted browsing interfaces. In order to deliver these features at scale, high performance query systems are required. We begin by describing the performance requirements of *pre-population*.

3.2.1 Pre-population

Pre-population is when areas of the interface are filled with all possible values at initial load, and then filtered down as and when selections are made. This differs from the initial experiments of mSpace which conformed to the Miller Columns[4] technique, where browsing was strictly left-to-right, where columns remained empty until selections were made. For example, in an mSpace of places to go in Montréal (see Figure 3.1), the Neighbourhood and Place columns are not populated, because no selection has yet been made in the Category column. In this particular layout, the user makes a selection in Category, which populates the Neighbourhood column. The user must then make a selection in the Neighbourhood column in order to populate the Place column.

A downside of this approach is that uses cannot see any neighbourhoods or places until they have selected a category. Thus, in order to enable users to get a quick overview of

[4]Mark S. Miller invented "Miller Columns," a filesystem browsing interaction technique used at Datapoint and NextStep, and now used by the OS X Finder application.



mSpace + Google Maps meets Open Guides

Welcome to the mSpace version of an Open Guide for Montreal. Open Guides

OpenGuides™ is a network of free, community-maintained wiki guidebooks to places around the world. Anyone is free to contribute, whether it's by writing new articles or editing the articles that we already have.

As an experiment in exploratory search, we've put an mSpace interface onto this Open Guide for Montreal. We've also added **Google Maps** so that a place added to the Open Guide can be quickly plotted on the Map. As long as you have a postal code for the place you enter, we can pretty much plot it on the map

In the Info view you can see any comments that have been added about a

Montreal Openguide Explorer. Part of the mSpace network. Click here to visit the CHI Schedule Explorer.

Waiting for montreal.mspace.fm...

FIGURE 3.1: An mSpace of places to go in Montréal, illustrating columns that are not pre-populated.

the space, and to reveal relationships in more than one facet at once, we explored the addition of a pre-populated model. In a pre-populated mSpace, all columns are filled with all possible values until the user makes a selection in any of the columns. Following a selection, all columns to the right of the selection are then filtered to only the selection that the user has made.

Pre-populating every column brings about a performance cost. Specifically that much more data is now required by the interface at both the start of the interaction, but also whenever a user clicks. Thus, the requirement on performance is raised significantly.

3.2.2 Backwards Highlighting

A consequence of adding pre-population is that a one-sided relationship between selections and action has been created. Specifically, that a click filters all columns to the right of the column of the selection. However, there is no relationship to any of the columns to the left. A possible interaction could be that all columns get updated when selections are made, however this sacrifice the user's original path, and the decisions they made, and thus in order to change their mind and make alternative selections, they would have to manually deselect their selections before all other possible selections are shown. However, we do want to be able to show possible selections in the left columns that relate to the selections that have been made, whereby it highlights the items that relate to the current selection. One of the challenges to this approach, as with pre-population, is that additional queries are required, in order to support it, and thus the requirement of high performance is raised again.

3.2.3 Large Scale

Another factor that pushes performance requirements is the need to support browsing over large scale datasets. As the number of records that the dataset uses increases, the overhead of the query tasks increases, and the performance drops. However, given that our aims are for users to be able to utilise multiple datasets, the requirements do therefore include large scale support. This means that our solutions must provide performance as the interfaces increase in size.

As noted in the above sections, there are also several features that are required by our solutions, in addition to supporting high performance querying, and thus we require solutions to both of these issues simultaneously.

3.3 Problem 3: How to Enable Users to Make Connections Across Data Sources

In this section we will outline the problems with browsing across data sources, and the requirements from our solutions to solve them.

3.3.1 Opening up Silos of Faceted Data with Semantic Web Tools

The world wide web provides a platform that organisations can use to publish data and associated visualisations. This enables a user to go to their website and see the organisations data visualised in a way that the organisation has chosen. However, given that there are multiple different organisations publishing data, there is an opportunity for use of data from multiple sources at once, in order to make more informed decisions.

Thus, in order for a user to be able to use data from multiple sites, each individual “silo” of data must be opened up so that it can be used in conjunction with other sources of data. Specifically, we have scoped our research such that our tools work with the Semantic Web, because it provides a domain-agnostic approach to publishing and consuming data. Therefore we require a methodology for opening up a source of faceted data so that it can be used on the semantic web.

3.3.2 Shared Facets and Instances

When multiple interfaces have facets that contain the same type of data, it is possible to use those facets as a link between the interfaces, to "pivot" across them. There are two key points where pivoting across facets may be desirable. Firstly, when two interfaces share a facet, and secondly, where an entity is shared over multiple interfaces.

Thus, in order to enable pivoting in faceted browsers, we require our interfaces to be able to determine shared facets and shared instances. As noted in Section 2.2, there are challenges in determining that two instances are the same, which is referred to as the "co-reference problem." The challenge is determining whether two different entities are the same. This challenge is particularly difficult when multiple sources of data are used, as is the case when sharing facets across interfaces. For example when one interface refers to "Chocolate Chip Muffins," while another interface refers to "Choc-chip Muffins," those are the same thing, and thus should be unified as a shared instance. That way, any user that selects "Choc-chip Muffins" will be able to pivot across to the data associated with "Chocolate Chip Muffins."

We therefore require that shared facets and instances can be determined and associated within our solutions.

3.3.3 Pivoting across faceted interfaces

In order to enable users to be able to browse multiple sources of data, we require that our user interface supports pivoting. Specifically we refer to the ability to browse from one independent domain to another via a faceted browser. In order to enable this, the above requirement of being able to specify shared facets and shared instances can be utilised, so that a user can use the shared facet as the pivot between the interfaces. For example, by pivoting from a nutritional interface to a grocery shopping interface via shared facets of "Food" and "Product," which contain shared facets like "Chocolate Chip Muffins." Thus, our requirement is a user interface technique that can be used by a user so that the faceted interface can utilise the shared facets and shared instances to enable pivoting by the user across interfaces.

3.4 Conclusion

In conclusion we have outlined a number of key requirements that our solutions will work towards solving. Specifically we have outline the following requirements:

1. A UI for describing datasets

2. A UI for creating faceted interfaces
3. A UI for browsing linked data
4. A faceted abstraction of a dataset
5. Cross-interface faceted abstractions
6. Pre-population of interfaces, with high performance
7. Backwards highlighting of interfaces, with high performance
8. Large scale amounts of data in a faceted interface, with high performance
9. Opening up silos of faceted data onto the Semantic Web
10. Determining shared facets and shared interfaces
11. Pivoting from one faceted interface to another

In the next chapter we discuss our solutions to the above problems.

Chapter 4

Solutions for Exploring Heterogeneous Data

In this chapter, we present our solutions to enable users to explore heterogeneous data through an integrated framework. Specifically, we focus on providing solutions to the three problems identified in the previous chapter: 1) how to enable people to explore heterogeneous data; 2) how to provide users rich features for browsing without sacrificing the performance of that makes them want to use it; and 3) how to enable users to make connections across data sources.

In order to provide solutions to these problems, in Section 4.1 we break down the process of how faceted exploration interfaces are created and introduce our solutions. Then, in Section 4.2 we detail our framework's design which we use apply our solutions. The remaining sections describe our contributions that enable users to explore heterogeneous data, specifically our: data picking interface, mSpace faceted browser framework, Facet Ontology, mSpace maker, Facet Portal, semi-automated facet semantic alignment service, and pivoting faceted browsing. Finally, in Section 4.10 we summarise our contributions.

4.1 Outline for Exploring Heterogeneous Data

In order to describe how we support and enable users to explore heterogeneous data, we outline four stages in which a user creates a faceted browser and how our contributions support them.

1. **Pick data to Browse**: A user first uses the **Data Picker** tool to select data to include into in a faceted interface. In order to simplify this process, the picker's interface is designed to minimise the information that the user is exposed to, provide exemplar data to the user, and display a preview of the data selected in

a live faceted browser. In more detail, the data picker automates the process of decoding linked data, determining the information within it, and previewing that information in a live faceted browser so that the user is presented with immediate feedback on what their resulting faceted interface will look like. After a user has selected and confirmed the data to be included in the browser, the user then inputs a SPARQL endpoint address or URI of some linked data. The data picker then queries/gathers the required data and asserts it into a knowledge base. The data picker then extracts the class and predicate names from the selected knowledge and presents them to the user in a nested viewer. As the user explores the nested data, any fields that contain literals are extracted as facets in a live preview mSpace. The user can then remove facets from this preview (and thus the final mSpace), this allows them to remove irrelevant facets. When the user confirms their selection again by entering their e-mail address with their selections. When the facet browser is created the user is e-mailed.

2. **Encode a user's selections formally**: We contribute **Facet Ontology** to encode a user's selection formally. It is transparent to the user and provides them with added value because it mints their selections as new linked data. This linked data can be used to inform exploratory tools by providing an abstraction of the chosen dataset. The Facet Ontology is a collection of terminology that defines how data should be queried and filtered before the data is available to a faceted browser. The Facet Ontology defines a "facet collection" which describes the data that the user selected. The collection contains the data sources, made up of URIs of RDF files, and URLs of SPARQL endpoints, and a number of facets. Each facet collection has one "first order" class, which represents the type of the "record" (for example, a faceted browser about classical music might have "Piece of Music" as the record type). The facet collection also defines "connected facets" as attributes of that first order facet (for example a "Composer" is a connected facet). Connected facets are connected via a chain of predicates. In our example, we create a connected facet of "Composer Birthplace" which is defined as the chain of directed predicates: "composer"; "born-in"; where backlinks are supported. Each facet is defined by a label and a number of additional definitions that enable the data to be filtered before being shown to humans in the faceted interface. These filters can define limits of the data collected (for example to only collect values greater than or less than a specified value), and to extract data from the raw fields (for example to perform a regular expression over ISO8601 dates to extract a year). In order for a faceted interface to be created from the facet ontology definition, a tool called the "mSpace Maker" can be used. The mSpace Maker has no interface component, and as such, the facet ontology definition that is created by the data picker from the user's selections is automatically submitted by the data picker to the mSpace Maker.

3. **Automate the creation of an mSpace**: We contribute the mSpace Maker tool to automate the creation of an mSpace over the data abstraction. The tool provides a minimal form that prompts a user for their email address and the location of their facet ontology definition. In the data picker, the facet ontology definition location is automatically provided, and transparent to the user. The mSpace Maker works in the background, gathering data, creating a database and configuring an mSpace faceted interface using the user's selected data. The user is e-mailed the URL of the new mSpace interface upon completion. The mSpace maker works by using the predicate chain and class definitions in the Facet Ontology description to generate SPARQL queries against the data sources. The returned data is then saved into a relational database, which has a schema that is generated based on the relationships in the original data sources.

4. **Automatically optimise the interface**: In order to deliver an optimal experience with a plethora of proven features , we create a database on-demand that is normalised based on the structure of the data that is picked by the user. We then perform a number of optimisations that automatically create compound indexes across the database tables which are optimised for the queries that mSpace performs on the database. The optimisations have been developed to provide optimal performance without any input from the user, so that they can generate an optimally fast interface without possessing any technical skills.

We illustrate the interactions of our contributions with the sequence diagram in Figure 4.1.

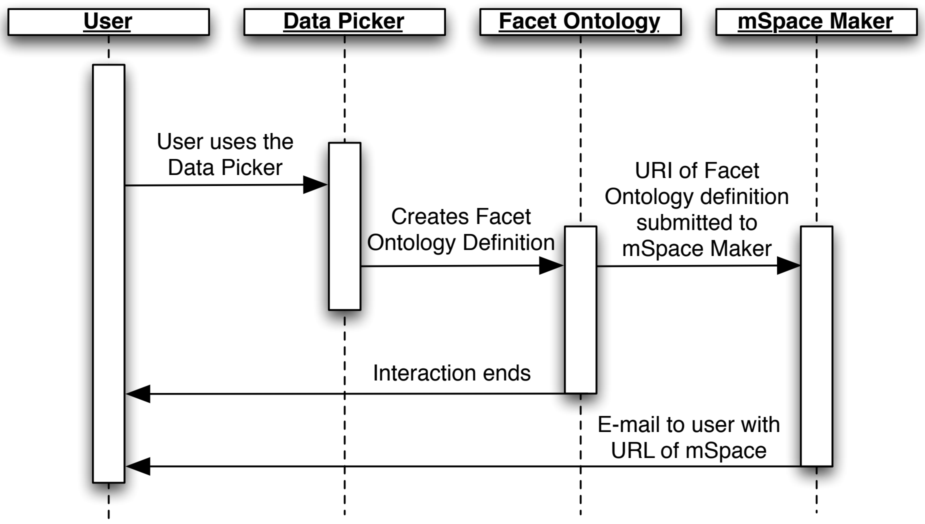

User
Data Picker
Facet Ontology
mSpace Maker
User uses the Data Picker
Creates Facet Ontology Definition
URI of Facet Ontology definition submitted to mSpace Maker
Interaction ends
E-mail to user with URL of mSpace

Figure 4.1: A sequence diagram showing how a user interacts with the components of the system.

We have delivered these four contributions because they enable users to create rich and high-performing interfaces based on the personal choices of the user, over arbitrary datasets, through use of an easy to use selection tool. In addition, by using a formal definition of the data with Facet Ontology, we allow facet selections to be reused by anyone else. For example, to create additional (non-mSpace) browsers of those facets, or to use the definitions as a basis for alternate facet ontology definitions. In the next sections, we describe each part of this approach in more detail.

4.1.1 Data Picking Interface

A key challenge in creating a faceted interface over a data source is being able to explore the source data, and accurately describe the semantic relationships between data that should be shown in the faceted interface. In order to enable all users, and not just those that have experience in the Semantic Web, to be able to pick data from a knowledge base, we contribute the Data Picker tool. Our tool accepts an entity URI or a SPARQL endpoint URL as its input, and displays data within that knowledge base in minimally technical way. Specifically, we render the data as a scrollable column, as it would look if it were included in a faceted browser. This approach therefore shows users what they are adding to an interface by accepting a particular class of data into their browser, and avoids users having to see and understand concepts such as URIs or ontologies. The Data Picker takes a lightweight approach known as "graph walking," where users select predicates and objects from the RDF graph in a knowledge base. Thus, our approach to picking data shows classes of available data to the user. The user then picks a class of interest, and all datatype properties (i.e. all of those that lead to literals) are rendered as a column, and added to the preview faceted interface. The user can then select any they wish to exclude. The preview faceted interface therefore works by including all possible facets; the default-include nature of the interface is intentional, as it allows users to browse through large scale data sources adding a number of facets without having to select them all individually. When the graph walker encounters any object properties (i.e. predicates that lead to other objects, such as "composed by") they are then shown in the graph walker, and all literals of the linked object class are then added to the faceted browser. The exploration is then a recursive repetition of the above actions, until the user decides they have all of the data they require. The data picker then creates a Facet Ontology definition of the data, and submits this to the mSpace Maker, so that the user can have access to a faceted interface straight away, without the need to submit their definition to the mSpace Maker or handle the definition URL in any way.

4.1.2 mSpace Faceted Browser Framework

In order to provide users with an intuitive exploration interface for multi-dimensional data, we contribute the mSpace Faceted Browser Framework. mSpace is an in-browser exploration framework that enables high-dimensional, large scale datasets to be browsed with high performance. Specifically mSpace is a faceted column-browser that enables a large number of facets to pulled into a view, and filtered by a user. In this thesis we evaluate the performance considerations required to enable a faceted browser to support scalable browsing of large scale datasets, while supporting a variety of interactions that stretch the performance requirements of the query engine.

4.1.3 Facet Ontology

In order to inform the faceted interface which data it should be populated with, a description of an abstracted view of that data is required. We contribute Facet Ontology, which is used to describe the structure of data that should be used to populate faceted browsers. Facet Ontology is an OWL ontology that allows RDF classes and predicates to be specified as relationships between data that are required for use in an interface. Facet Ontology allows transformations of data to be specified, so that when data is harvested, it is filtered through different algorithms before it is rendered and seen by the user. For example, a date facet might be filtered to just a "year," in order to create a Year facet. The use of Facet Ontology means that a model of a domain can be shared, because it is written in RDF, and therefore has a URI, which is resolvable using the methodologies of Linked Data. By sharing descriptions of models, they can be extended and used as a basis for further interfaces of the same, similar or overlapping domains that use the same data source(s).

4.1.4 mSpace Maker

To translate a Facet Ontology description into a working faceted browsing interface, a data extraction tool must be used. Our contribution, mSpace Maker, performs this task by creating faceted interfaces using the mSpace framework. mSpace Maker uses the URI of a Facet Ontology definition, it extracts the knowledge from data sources and populates an mSpace database with that knowledge. A basic mSpace, ready to be customised and styled, is then deployed over that mSpace database. The submitter of the mSpace to the maker is then e-mailed with the URL, and is able to browse the data specified in the Facet Ontology definition. The mSpace Maker is extensible so that additional data transformations can be added, that conform to third-party ontologies.

4.1.5 Facet Portal

One of the benefits of using Facet Ontology to describing data spaces, is that the definition can be shared and built on by others. In order to simplify this process, we contribute the Facet Portal service. This service offers a registry of Facet Ontology definitions, which it indexes so that users can search for interface definitions that match their domain of interest. The portal registry also allows faceted interfaces and data pickers to query the indexes in order to find overlapping domain models, to aid with extending existing models, and pivoting from one model to another.

4.1.6 Semi-automated Facet Semantic Alignment Service

Our alignment service contributes a lightweight mechanism for semi-supervised co-reference of entities from multiple sources. Entities from data sources described in Facet Ontology definitions are aligned using the Alignment API (Euzenat, 2004) (originally designed for aligning OWL classes, but re-purposed for our faceted alignment). Alignments are created that map between two facets from different data sources, where there is overlap in the instances. The alignments are then used to create the pivot between two data sources. For example the Product facet in a supermarket interface contains "Cauliflower," "Carrot," and "Broccoli," and maps to those instances in the Food facet of a nutrition interface.

4.1.7 Pivoting Faceted Browsing

We contribute pivot support for faceted browsing, so that users of a faceted interface can search for additional domains, and drag in facets from those domains in order to filter their current browsing interface using those facets. Support for pivoting in mSpace provides a lightweight interaction technique that utilises the information on the facets held in a facet portal, so that users can take advantage of facet definitions that exist, and browse across domains, through pivots. When alignments to a faceted interface have been created, the facets from the aligned dataset are exposed, and users can drag them into their mSpace slice as they do with facets of the original domain.

In the next section we describe these approaches in more detail, against the background of the research challenges identified in Chapters 2 and 3.

4.2 Approaches to Research Challenges

By contributing the work discussed above, we enable users to explore heterogeneous data and address the following four challenges:

1. **Building a generalised model.** In order to deliver a domain or area of a domain that is of interest to the user, a model of the data space has to be constructed.
2. **Developing an appropriate User Interface.** Given an appropriate model of a data space, an exploratory interface, suitable to allow a human to increase their knowledge, has to be developed that can use the model.

3. **Delivering performance to the exploratory interface.** In order to provide an acceptable user experience, an exploratory faceted interface must not freeze on the user, or take too long to answer queries. As data spaces get more complex and large, the challenge of delivering acceptable performance increases, and has to be addressed appropriately.

4. **Extending performance back to re-usable tools.** The final challenge area of this chapter is how to apply the approach to the above performance challenge so that users themselves can utilise this performance, without having to have the knowledge that we have developed ourselves. Otherwise users would not be able to create their own interfaces over data and our approach would be limited.

In the following subsections we describe the above challenges in more detail, and outline where contributions are made, starting with the model:

4.2.1 Building a Generalised Model

We present two contributions to building generalised models of abstract data spaces. First, the mSpace Model, which maps a single semantic knowledge base to a database schema, and provides an mSpace over that database; and second, Facet Ontology, which generalises and formalises the basic concepts of the mSpace Model, and applies it to multiple knowledge bases and arbitrary RDF linked data on the Semantic Web.

The mSpace Model is designed to describe the structure of information in a knowledge base, so that an abstraction of information of interest to users can be queried. This enables faceted browsers, such as mSpace, to generate queries so that the knowledge base returns metadata filtered by users during an exploration session. The inspiration for creating a generic faceted browsing framework was CSAKTiveSpace (mc schraefel, Shadbolt, Gibbins, Harris, and Glaser, 2004), which presents heterogeneous data in the domain of computer science research in a faceted interface. We hypothesised that the proven usefulness of CSAKTiveSpace in the computer science domain could be applied to other domains. Hence we presented a novel general approach that enables data of any domain to be described so that the data can be used to populate any faceted interface framework. Thus enabling the model to be re-used and re-purposed for further domains and datasets.

The mSpace Model uses the same knowledge storage architecture that CSAKTiveSpace used. Specifically, it uses a Semantic Web triplestore, queried initially in RDQL, and now SPARQL, which is populated with an RDF representation of the metadata of a particular domain. In order to populate the triplestore with metadata, the data is first converted into RDF, and the particular ontology (or schema) that the data uses is then directly referenced by the queries. The benefit of this approach is that there is no requirement for *a priori* definition of the ontology or the vocabulary used, unlike in a database system. To this end, the mSpace Model was developed to encapsulate a definition of how to query a knowledge base to filter metadata and permit faceted browsing.

Additionally, the mSpace Model supports use over any source of RDF that can be asserted into a knowledge base, and as such, several key features are important to support, the first of which is referred to as "multi-hop." In a classical music example, the metadata is connected in the graph through a single triple (or "hop"), whereas some data is normalised such that in order to query the data required to populate a column, a chain of multiple triples has to be followed. An example of this is that it is possible to browse the classical music dataset by the year of death of the composers, by making selections in the *composer death year* column. The definition of the column's data is not through a single triple to the goal object, instead the required data pattern is of the two triple form:

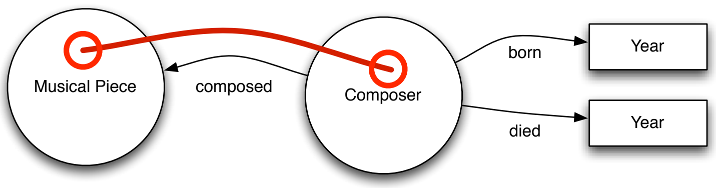
```
<piece>         <music:Composer>        <composer>
<composer>       <music:YearDied>        <year>
```

The ontological graph structure that the above triples signify is illustrated in Figure 4.2, which shows how three definitions of facets are defined. The thick red lines show the "multihop" definitions for three facets. The first, illustrated in Figure 4.2(a) shows the relationship between the musical pieces and their composer, and this relationship is used to define the Composer facet. In order to define facets for the composer's birth and death years, illustrated in Figures 4.2(b) and 4.2(c) respectively, a "multihop" definition is used.

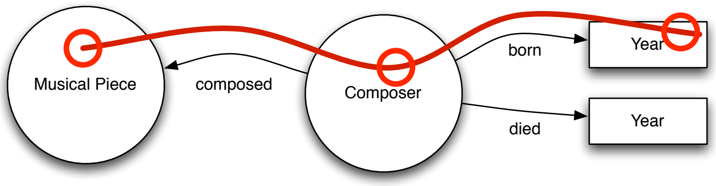
Multiple predicate relationships require the model pattern to support an ordered list of named predicates to match against, or "multi-hop" to, with the interim classes (in the above example, "composer"), remaining anonymous and invisible to the user, unless another separate column has been defined describing it. This is likely to be the case with RDF taken from the Semantic Web, as ontologies are frequently highly normalised in order to provide the most useful amount of knowledge and inference as possible, and therefore this feature support takes the framework closer to being able to work with any RDF and not just a dataset that has been carefully curated for the purpose of browsing using the mSpace interface.

The cost of marking up data using an mSpace Model is outweighed by the benefits of having a human-explorable abstraction over a dataset, and the ability to then create a faceted interface using that abstraction.

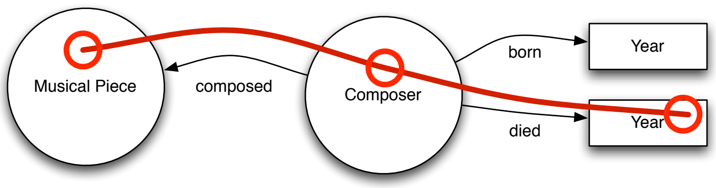
As described above, in order to provide the benefits of the mSpace Model to other faceted browsers, and to formally specify the mSpace Model's directives in an ontology, we contribute Facet Ontology.



(a) Single predicate relationship between a Musical Piece and its Composer.



(b) Two-predicate "multihop" relationship between a Musical Piece and the Birth Year of a Composer.



(c) Two-predicate "multihop" relationship between a Musical Piece and the Death Year of a Composer.

FIGURE 4.2: Illustration of a sub-section of a metadata ontology structure for classical music, showing the relationship between Musical Piece, its Composer, and the years of birth and death of the composer.

In addition to being more formal an open, Facet Ontology functionally adds to the mSpace Model the ability to define transformations and limits on data before they are added to the faceted interface. Specifically, Facet Ontology defines transformations to extract data according to specified patterns. For example, a facet can be defined that parses the data as an ISO8601 date, and extracts the year from that date. This kind of transformation is useful, because it enables the interface to break up the raw data into multiple facets, because in some datasets, a decade or year facet may be more useful than a date or month facet. Likewise, limits can be applied to facets, for example to place numeric bounds on valid information, if the interface shouldn't contain information with dates in the future.

We proceed by describe the challenges faced in developing an appropriate User Interface.

4.2.2 Developing an Appropriate User Interface

The User Interface for mSpace was designed to function with lightweight interactions, to enable users to both explore a data space that they were familiar with, as well as learn about a domain that they are unfamiliar. The latter case in particular benefits from lightweight interactions, as they allow users to see connections between items that they are unfamiliar with, in quick succession. For example a user learning about classical music can quickly click on a number of centuries sequentially to see the composers from each century, and thereby get a sense of the space in a small amount of time. Alternatively, if a user was required to use a more heavyweight interaction such as using a typical *advanced search* interface, it would take more actions and more time for the answers to appear, and the cost to the user of gaining that knowledge is higher.

Our lightweight approach is similar to that of faceted browsers such as Exhibit, which update their results in real-time. There is also a clear difference in place between web sites that can update in-place, and those which require a full page refresh to update. In-place updating is preferred over full page reloading, as it retains the context of the information and is therefore easier for users to follow what has been updated, and can be aided through cues such as appropriately placed loading icons.

Such an approach aids users in the exploration of a data domain, and enables information to be quickly triaged. In order to support a realtime reactive interface, the query backend must be optimised to return responses quickly. Optimisation of the backend incurs a cost, in terms of engineer time and hardware costs.

The next subsection describes how we approached the problem of getting performance to our initial deployments.

4.2.3 Delivering Performance to the Exploratory Interface

As the amount of data loaded into a knowledge base increased, the response times of the queries also increased. Thus, in order to support the fast-loading required to support a responsive UI, performance improvements to the querying subsystem were required.

Our pilot studies showed aligned findings with previous research (Nielsen, 1997) which showed that users leave web sites when pages take more than a number of seconds to load. Likewise with our faceted browser, if the browser did not update its interface in response to a click after a number of seconds, users either refreshed the interface, clicked other items in an attempt to "wake up" the interface (firing off more queries, and thus adding to their wait time), or simply stopped using the interface entirely. Such an experience results in users losing confidence and interest in the interface, and thus it is important the performance is prioritised in the design and implementation of the system, in order to not only stop users from leaving the system, but also to fully engage users with the content that the interface is attempting to help them access and explore.

Related work in exploratory interfaces also encounter performance issues, and can be broadly split into three categories of attempts to deal with performance problems:

1. Require less of the query subsystem, by removing features.
2. Accept performance problems, and warn users that slowness will sometimes occur.
3. Engineer fixes to performance problems.

One approach to increasing the perceived performance of an exploratory interface is to reduce the processing requirements of the interface. In particular, this can be achieved by turning off retrieval features within a faceted browser. For example, a common feature of a faceted browser is to show how many documents (or artefacts) match the given value of a facet (known as *cardinality*), in order to aid the user in knowing the size of the result set if they filter by that value. Calculation of this figure can be expensive, and becomes increasingly so when the number of possible values on screen at any time is large. Therefore, by removing the cardinality feature, the perceived performance of an interface can be improved.

Alternatively, in some interfaces the processing-intensive features are necessary for the users, and cannot be removed. In some cases performance issues can be simply accepted as the way the system is. In particular this can work when usage of the system is a requirement of users (for example when its results allow users to do their job), and when users receive training on the system, in order to manage their expectations of the system. Similarly, research prototypes have often been labelled as such, where the requirements of the project are focused on the collection and presentation of the metadata, rather than delivering a user experience for the general public. Ideally, performance issues can be solved through engineering, where possible.

As our key investigations require that users utilise the interface to explore domains that they are unfamiliar with, we are not in the situation where users are required to use the system or where they know what they are looking for; our intention is to present an interface to users whereby they can explore a new area quickly, learning more about the interface as they explore, thus requiring responsiveness at a level where the users can make selections, undo selections, see results and process changes in real-time. Therefore it is important that our faceted interface is responsive enough to allow users to perform such interactions, and engineering the interface for real-time responsiveness is crucial.

In the final subsection, we describe how performance from our initial deployments was translated to re-usable tools.

4.2.4 Translating Performance into Reusable Tools

One of the major performance gains in our interface development was through the use of optimised database normalisation. In order to enable this performance gain for all interfaces, we abstracted the ability to create optimal table layouts into the mSpace

Maker tool. Thus, any data imported using this tool is designed to take advantage of the performance gains that were realised in the work to date, automatically.

This approach is similar to that of exhibit, which is a tool to allow anyone that downloads it to be able to create a faceted interface over tabular structured data. Our approach is similar in that our performance gains are available to anyone that uses our code to load their RDF data into a database.

Such an approach has the benefit that the approach that generated performance gains is now re-usable through the tool. Thus, in future, anybody that needs to design a faceted browser can use our tool to get the high performance levels that our approach delivers.

The benefit to this approach outweighs the cost of producing the tools, because it enables future work, both our own and others to benefit from the performance. Specifically, in our test cases we require multiple mSpace faceted browsers to be created over a number of different data sets, and as we have created a tool to create optimised databases for these browsers, the time and effort required to generate these additional mSpaces is lowered. Similarly, third parties can also benefit from the tools, which has the benefit of raising awareness of the tool, which allows us to test our interface on additional data without having to perform the tasks for this ourselves; by presenting an easy-to-use desirable tool for exploration, we get users coming to us, with data that we can analyse, without having to canvas for test cases.

In order to confirm that our approaches to the research challenge of providing exploratory interfaces over data sources were successful, we tested them on a series of data sources, using experimental interfaces. The scope and environment of the experimental interfaces were set up such that the early experiments were concerned with predictably structured data, and we incrementally opened up the scope of the work such that the predictability and level of structure of the data expanded in subsequent experiments.

The core concepts tested concerned the exploration of structured data sets, the merging of heterogeneous data and post-hoc structuring of unstructured data, necessary to facilitate enhanced exploration. Our contribution is in the form of the *mSpace model* definition, that allows data to be described in a manner that allows a dynamic multi-faceted browser to be dynamically created based on the definition.

4.3 Automated Interface Creation with the mSpace Framework

In this section we describe our approach of automating the creation of a classical music interface, based on an abstract model over a data source.

In order to address the challenge of creating an abstract model over any data source, and building faceted interfaces over those models, we tested a number of a experimental interfaces over different domain areas. The first test case utilised well-structured data on classical music. By "well-structured" we mean that the data is available in a tabular format that allows the extraction of named fields (attributes) that relate to subjects. In this case, we have data on pieces of classical music, with fields such as *Piece Name* and *Composer*, with values such as *Fifth Symphony* and *Ludwig van Beethoven* (respectively) for example. Classical music was chosen for this test case as it is a domain to which there is a large amount of structured data freely available, and it is also a domain to which many people are familiar, but few are experts. This makes it an ideal dataset to use to test out how users' learning accelerated or was aided, because of the exploratory interface enhancements.

In terms of structure, the classical music data we used was taken from various sources, and the schemas of these sources were not well aligned and some had different syntax forms. For example, some preferred naming a composer as "Ludwig van Beethoven," others as "Beethoven, Ludwig van" (this problem is known as co-reference and is discussed in detail in Section 2.2). It was possible however, to piece together several sources of data to match up the overall sets. The original source of the information was the metadata taken from a collection of MP3[1] music files, which use the ID3[2] metadata format, a fixed and limited schema to describe attributes of the music, such as *Title*, *Artist* and *Year*. This presents numerous challenges, from syntactical to semantic. The syntactical problems are that the metadata is of a fixed length, and provides no mechanism for disambiguation or unique identification. Semantically, there is no provision for detailing extra metadata fields, such as the names of the performing orchestra, or the day of the performance (the only date field is *Year*).

Sources of data from classical music sites on the world wide web were used to augment the music metadata; dates of birth and death of composers, for example, were taken from the web, and string matched with the names of the composers from the music metadata, with non-matches fixed by hand. Then a source of start and end date ranges for music eras (such as *Romantic* and *Renaissance*) was identified online and connected into the dataset, leading to the inference that if a composer worked during the date range of an era that the work is of that era; in some cases this is incorrect, but it is good enough for the intent and purpose of this test case.

Pulling this data together was enhanced through the use of Semantic Web tools; the use of a triplestore(Rusher, 2001) allowed data to be pulled in on an ad-hoc basis without first (or at all) defining a schema. This built on ideas utilised originally in the CSAKTiveSpace project(mc schraefel et al., 2004), which pulled together information from a number of websites about computer science research in the UK, and presented it on a single exploratory interface. The interface was fixed to the domain of computer science research,

[1]MP3 History: `http://www.iis.fraunhofer.de/EN/bf/amm/mp3history/mp3history01.jsp`
[2]ID3 Metadata Standard: `http://www.id3.org/`

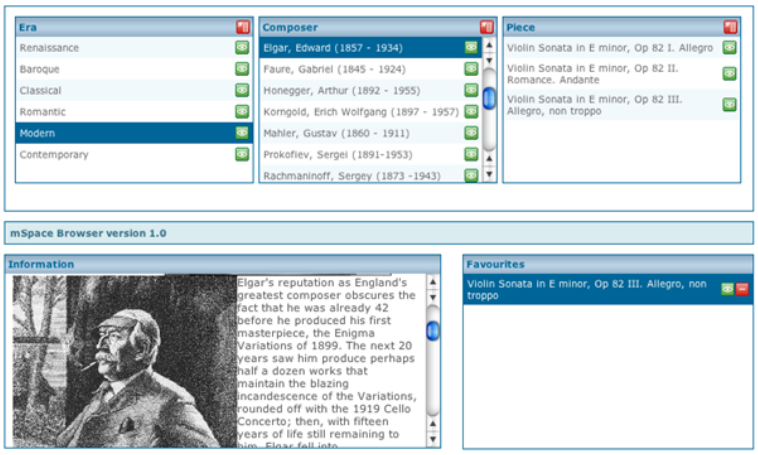
whereas our work on the *mSpace model* expands on this idea by allowing data from any domain to be described and brought together on one interface.

As the data was entirely generated and controlled by us, the model can rely on the data fitting a specific (Semantic) pattern of RDFS classes and subject, predicate, object triples. A model that allowed the description of these patterns was created, with the interface querying against this model to retrieve appropriate results.

This test case extends the hand-rolled domain-specific approach to semantic querying interfaces as used in CSAKTiveSpace, into a generic framework that can be configured dynamically, to apply to a specific domain or cross-domain data set. As a step towards automation, given a highly structured data set, such as the classical music data, an interface can be applied through the creation and assertion of description metadata that conforms to the *mSpace model* ontology. The metadata of the model informs the query generator, in the backend, the patterns that the data is made up of, relative to a fixed point (a particular class), known as the goal column. The model describes attributes as *dimensions*, which are represented in the interface as columns, in which the user can make selections, to filter the rest of the exploratory interface, see Figure 4.3. In the classical music example, the goal column is the class of pieces of music, which are described in the model only in terms of their RDFS class. Other columns are then described by how they, on the data graph, connect to classes of pieces of music. In the case of the definition of a composer, they follow the triple pattern:

```
<piece>    <music:Composer>    <composer>
```

This is mapped into the *mSpace Model* ontology, and repeated for all of the columns required, with the resulting RDF asserted into the same knowledge base of the data to be explored.


Era
Renaissance
Baroque
Classical
Romantic
Modern
Contemporary

Composer
Elgar, Edward (1857 - 1934)
Faure, Gabriel (1845 - 1924)
Honegger, Arthur (1892 - 1955)
Korngold, Erich Wolfgang (1897 - 1957)
Mahler, Gustav (1860 - 1911)
Prokofiev, Sergei (1891-1953)
Rachmaninoff, Sergey (1873 -1943)

Piece
Violin Sonata in E minor, Op 82 I. Allegro
Violin Sonata in E minor, Op 82 II. Romance. Andante
Violin Sonata in E minor, Op 82 III. Allegro, non troppo

mSpace Browser version 1.0

Information
Elgar's reputation as England's greatest composer obscures the fact that he was already 42 before he produced his first masterpiece, the Enigma Variations of 1899. The next 20 years saw him produce perhaps half a dozen works that maintain the blazing incandescence of the Variations, rounded off with the 1919 Cello Concerto; then, with fifteen years of life still remaining to him, Elgar fell into

Favourites
Violin Sonata in E minor, Op 82 III. Allegro, non troppo

FIGURE 4.3: Part of the mSpace interface, showing the columns, which represent attributes in the data.

In terms of the robustness of the system, the prototype benefited from a large scale user concurrency load test when it was linked to by the popular UK news website "The Register,"[3] which in turn led to it being listed on the popular US technology news website "Slashdot,"[4] which is renown for its readership being large enough to break websites that can't handle the load, known as Slashdotting[5], which the prototype handled.

The system's interface was designed to support context during exploration, specifically so that after a user makes a selection, the unchosen alternatives are not removed from view, as opposed to typical behaviour on exploratory systems. This context allows the user to persistently see how they arrived at the current point, with the current results. It also allows the user to alter their choices, without having to page back through their other choices, and having to recreate them.

Our test case demonstrated that a data modelling approach can be used to create interfaces for a dataset for a domain, where the data has been marked up using RDF. The benefit to this approach is that we could gather data from multiple sources online, mark them up in RDF using a single ontology and have them appear in the interface. This is possible because our approach queries a triplestore based on the mSpace Model, and we are free to keep adding data, from multiple sources, to that triplestore.

The test case proved that it was possible and useful to create a framework that can be used to browse generically over a dataset. We showed that through a responsive interface, users that are unfamiliar with a domain (in this case Classical Music) are able to quickly explore and learn about that domain, and in this case, use that ability to find out what aspects of classical music they like, for example composers, eras, types of instrument and so on. A major contribution was the mSpace Model and associated framework that allows any knowledge to be abstracted into a faceted browsing mSpace interface. We developed a technique we call "multi-hop" whereby a facet can be related through another facet, where this relationship is present in the semantic data, which means that the interface is capable of handling complex relationships within knowledge, rather than relying on a flat set of literal values for each metadata, as is typical.

Our implementation showed that the mSpace Model worked for enabling browsing over a particular domain. In order to expand the scope of the interface and enable it to support large datasets we explored the performance considerations for faceted browsers, which we detail in the next section.

[3]The Register: `http://www.theregister.co.uk/`
[4]Slashdot: `http://www.slashdot.org/`
[5]Slashdotting: `http://www.sentex.net/~mwandel/traffic/slashdotted.html`

4.4 Getting Data to the UI — Query Performance Considerations for Faceted Browsing

This section discusses the problem of getting data to the UI in a timely and scalable manner. During the previous mSpace test cases, the outcomes were often that additional features were added to the framework that either displayed information to the user, or allowed the user to filter the information displayed. This adds pressure on to the backend, to push more data to the user and allow more complex queries from the user, creating performance problems. In addition, larger data sets were introduced which added further performance and scalability challenges.

Specifically, the challenges are both technical and user-oriented. The technical challenges are regarding how to store and model data so that queries against the data return results fast enough for the user experience to remain good. Furthermore, it is important to prevent the "paradox of choice" (Schwartz, 2005) from affecting the user. The paradox is that by offering the user a larger set of options, the number of selections the user makes goes down, rather than up. The user-oriented challenges are how to demonstrate to the user that there is a lot of data to explore and choose from without overwhelming them and causing the paradox to occur.

In this section we describe test cases into triplestore scalability, specifically the limits on the amount of information that can be loaded into triplestores, and getting triplestores to achieve query response times suitable for supporting well-performing user interfaces. We describe two interface enhancements: Pre-population and Backwards Highlighting, and how their addition led to performance penalties that became show stoppers for the user interface, and how we delivered performance to support them. We then detail our experiences with different triplestore methodologies, specifically describing test cases where we used two different systems to attempt to support our user interface. We conclude with our contribution of a technique, *dynamic list paging*, which aids in the scalability of web-based exploratory user interfaces.

Performance: Related Work

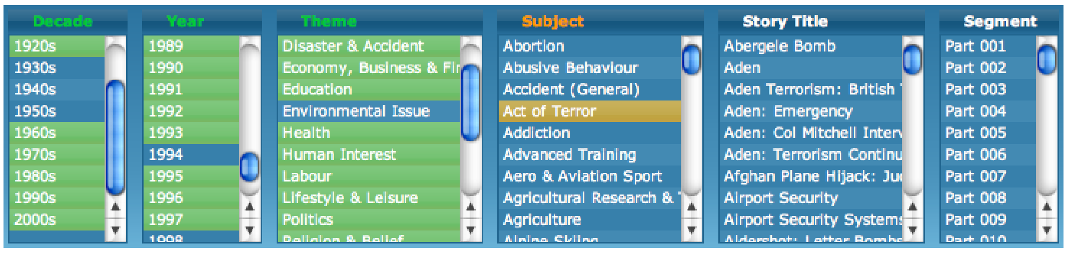
In the Semantic Web query space, one of the research pushes has been towards scalability of triplestores. Specifically this challenge can be broken down into the amount of total triples that can be loaded into a store, and how quickly queries return results from those stores.

There have been two distinct approaches to these problems, to store triples using already-available relational database systems, or using proprietary storage engines. Using the former approach, D2R Server (Bizer and Seaborne, 2004) presents a language for mapping a database schema to Semantic Web triples, so that a database can be queried using

SPARQL. This approach benefits from being able to store and index data using the storage and indexing mechanisms of the underlying database software, which may include clustering features, for example. The latter methodology uses storage engines specifically designed for use with RDF, such as *3store* (Harris and Gibbins, 2003), which has the benefit that indexing particular to the SPARQL queries can be used, although this technology is not as mature as database storage engines and thus isn't as optimised or featureful, although new storage technology such as in *4store* (Harris et al.) provides the ability to cluster the storage of triples, in order to increase ceiling number of triples that a knowledge base can hold.

Pre-population and Backwards Highlighting of Faceted Interfaces

mSpace was demonstrated at CHI2006, which led to a significant amount of user feedback, both with respect to the user experience, and the technical limitations of the back end. One key feature that was raised, and carried forward into our prototypes, is the concept of 'pre-population' of the mSpace columns. This means that all columns have data presented in them at the start of the exploration, and not as in 0the original mSpace concept of the columns being filled from left to right, only when the items were selected. When a user makes a selection on a column, those columns to the left will highlight all possible 'routes' (selections) that the user could have taken in order to select the item, referred hereafter as "backwards highlighting," see Figure 4.4 for a screenshot that shows pre-population and backwards highlighting. Columns on the right of the selection are then filtered based on this new constraint. We have found that this allows much quicker interaction with the interface, as users can view more of the available choices, therefore getting a better overview of the domain when they first visit the browser.



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 4.4: A screenshot of the mSpace interface showing backwards highlighting in green, from the user's selection in orange.

Adding pre-population and backwards highlighting vastly increases load on the backend: for every selection, the original mSpace design required one query, in order to populate the next column. Now, one query is required for every visible column, and these queries must be run every time a selection is made. Using the original querying methodology would simply not scale, when hundreds of users are trying to browse data and requiring that level of queries per selection, while expecting to be able to browse in real time. Thus, alternative mechanisms were developed to enable such support.

In order to formalise our model to enable definitions to be utilised and extended by third-parties, we developed our model into a more defined ontology, which we describe in the next section.

4.5 Facet Ontology: Enabling Users to Describe Faceted Spaces in Metadata

In order to achieve our requirement to formally describe a faceted abstraction over arbitrary data, we present FacetOntology. FacetOntology is an ontology that facilitates describing facets and their connections, from a RDF data source.

Facet Ontology provides a lightweight mechanism to enable RDF to be described for a faceted presentation and interface. Specifically, Facet Ontology provides a vocabulary for describing the human-readable attributes within RDF, and how to query for them. A definition using Facet Ontology gives a faceted browser a specification for the facets that exist for a given domain, preventing extraneous RDF from overwhelming the user with potentially off-topic data, by providing a bounded abstraction over arbitrary RDF. For example, in the shopping domain when making a facet definition for shoppers, we might want to include facets on "price," and "food group," while not including internal SKU reference numbers and UPC codes. On the other hand, we could additionally make a facet definition for internal use of the data that does include such information.

Facet Ontology achieves a lower overhead by having common default values for structural elements of RDF, based on best practices and commonly used ontologies: This means that a very lightweight description of just the RDF Class of a facet can be used, with support for additional constraints where the structure of the RDF deems it necessary. Otherwise, Facet Ontology uses defaults, such as using the standard `rdfs:label` predicate to find string labels of individuals in RDF, while allowing this to be overridden to suit a particular dataset. The ability to combine linked data that is so key to tabulator is now also key to Facet Ontology, specifically URIs of RDF documents can be used as data sources using Facet Ontology, and these are combined ahead of time on a server in order to allow the system to scale-up past a small amount of RDF.

4.5.1 Describing and Sharing Facet Definitions

The Facet Ontology contains a vocabulary that can be used to describe facets, whereby a facet is described with attributes and relationships that hold among these attributes. These attributes and their relations can be used to describe the possible behaviour of these facets in a faceted browser. Specifically, facets can be used to generate meaningful SPARQL queries given the elements used in the mark up the data source[6]. These

[6]These elements are defined in the original data source's ontology.

SPARQL queries are used by the faceted browser to aggregate data sources and render them as facets. This ontology has not been designed for a specific faceted browser, and instead describes a refined version of the faceted description from mSpace (described in (m. c. schraefel et al., 2005)), which we have attempted to simplify so that it is compatible with any faceted browsing approach.

We now formally introduce the components required by a Facet Ontology definition: Let D_1 and D_2 be two data sets which contain facets and entities, where $D_1 = \{f_{1,1}, \dots f_{1,n}, e_{1,1}, \dots e_{1,n}\}$ and $D_2 = \{f_{2,1}, \dots f_{2,n}, e_{2,1}, \dots e_{2,n}\}$, where $f_{1,1}, \dots, f_{1,n}$ and $e_{1,1}, \dots, e_{1,n}$ represent a sets of facets and entities, respectively. Let P_{D_1,D_2} be the set of data that pivots between D_1 and D_2, where $P_{D_1,D_2} = \{f_1, \dots f_n\}$ and $\forall d \in P_{D_1,\dots D_2} : f \in D_1 \wedge f \in D_2$. In order to illustrate our pivoting approach, we provide the following example: Given two data sets D_1 and D_2, where D_1 represents a large supermarket, and D_2 represents collection of nutrition data for different foods. Facets such as "product," "price," and "food category," are contained in D_1 and facets such as "food type," "calories," and "total fats" are contained in D_2. Both of these data sets contain pivotable data, where $P_{D_1,D_2} = \{food\ type,\ food\ group\}$. These pivots identify that both data sets share the entities such as "White Rice," "Tuna Steak" and "Wholemeal Bread."

The pivot data provides a link between data from D_1 and D_2, and enables queries on facets from the supermarket domain, to connect through the identified pivots from the domain of nutritional data.

In order to enable relationships between facets to be made, Facet Ontology describes two key categories of facet: a "first-order facet," and a set of "connected facets." We define these two categories below:

1. A first-order facet ($f1$), is described by its rdf class (using the `rdf:type` predicate).
2. Connected facets (fc_n), are described by how they connect in the data graph in relation to $f1$. This is modelled as a chain of predicates that are used to generate the SPARQL query that gathers the data.

A facet collection (F) is a set of facets, where $F \leftarrow \{fc_1 \dots fc_n\} \cup \{f1\}$, that can be used to create a faceted browser to explore a particular dataset of a particular domain.

For example, consider the simplified graph of metadata shown in Figure 4.5:

In the supermarket domain there are two classes, Product, which represents an item sold in the supermarket, and Food Type, which represents a categorisation of products. The classes are linked using the `ns:foodType` predicate, and both of these classes have various attributes modelled using different predicates. In our Facet Ontology definition for this data, the Product class is used to define the first-order facet, with the predicate `ns:foodType` being used as the predicate chain to define the Food Type facet. This

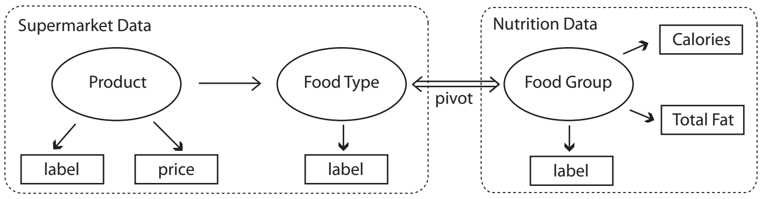


FIGURE 4.5: A simplied data graph from the supermarket products domain, and related information in the nutritional information domain, showing the Product class, the Food Type class, the Food Group class, and several of their attributes.

means that when the data gatherer (in this case, the mSpace Maker) queries the data sources, it creates the following two SPARQL queries to gather the data, where the variables `?label` and `?uri` gather the label and uri, respectively, of the items to be shown in the facet browser metadata columns:

1. Query to gather the first-order facet of Product:

   ```
   SELECT ?label ?uri WHERE {
          ?uri rdf:type Product .
          ?uri rdfs:label ?label
   }
   ```

   This query returns a table of URIs and labels that represent all of the items to populate the Product facet.

2. Query to gather connected metadata about the Food Type (where the variable `?firstorder` will contain the URI of the Product in the first-order facet:

   ```
   SELECT ?label ?uri ?firstorder WHERE {
          ?firstorder rdf:type Product .
          ?firstorder ns:foodType ?uri .
          ?uri rdfs:label ?label
   }
   ```

   This query returns a table of URIs and labels to populate the Food Type facet with, as well as the URI of first-order facet items (Products, in this case) that have this food type. The facted interface can use these URIs to make the link between the first-order items and the Food Type facet, so that when users filter on a food type, the products of this food type can be shown.

A set of optional filters are also available in Facet Ontology to further filter the queries that select the data. For example, by default the `rdfs:label` predicate is used to gather the labels of attributes, and this can be optionally changed, to suit the particular metadata of interest. Additional hints to for the interface, such as specifying a default numerical (as opposed to alpha-numeric) sort within a particular facet.

One of the barriers to entry of this approach is the creation of the FO definition. As such, we have developed a “data picker” tool that provides users with an intuitive interface for picking the metadata that they wish to be marked up using Facet Ontology. The tool works by querying a data set for all RDF classes, and a subset of the labels of individuals of that type. These are displayed, and the user can select a class which they want to explore further. All of the predicates that are joined to individuals of that class are then shown, again with a sample of their values. A user can then select which attributes they wish to be marked up using FO. This is a semi-automated approach, in that users do not need to see any FO RDF, but they do need to inform an interface which facets should show up in their faceted interface.

The need to aid in the manual marking up facets is not a new problem, even within the domain of the Semantic Web. Work by (Oren et al., 2006) looked at ranking facet quality, as an aid to automatically marking up facets, where their technique was formally proven to show an improvement of quality. Similary, AKTiveRank (Alani et al., 2006) presents a technique for ranking ontologies, using structural metrics. Such metrics could be used to inform a facet creation tool, in order to enable an increased level of automation.

In summary, the cost of creating a Facet Ontology is linear with the amount of facets in a domain - if a domain has 4 facets, a single First Order facet is defined by its RDF Class, and 3 Connected Facets are defined by how they link to the First Order facet. In a domain with a larger amount of facets, a larger definition is required, which will take more time to create.

There are benefits beyond faceted browsing to creating a Facet Ontology, specifically with respect to using the markup of Facet Ontology as a way to add discoverability to your data set. When a data publisher releases a new SPARQL end point or set of RDF files that represent their data, they can mark up the important fields of interest using Facet Ontology, and specify the data sources as their newly released files. This brings together a definition of “what” a piece of data is, and “where” to find it, since it’s beneficial to say that a new data source on Music has been released, and to provide an ontology with it, but it’s even better if you can point out what fields are human-readable and suitable to browse, and where to find the data containing them.

There are two key specifications of FacetOntology that work slightly differently: a standard specification and an advanced specification. The standard specification requires a simpler description of data, and is useful and usable for most cases. In a case where the data does not fit the standard specification’s model, or the data requires pre-processing and transformation before being using in a faceted browser, the advanced specification can be used.

In an ideal software engineering situation, the standard specification would be implemented as a template that generates an advanced specification first, enabling the query generator to work off of the advanced specification only.

The advantage to using the *standard specification* is that it is simpler to understand, compared to the verbose *advanced selection*, which requires *a priori* knowledge of the data types in use, and any issues that need to be fixed.

A serialisation in Turtle of the Facet Ontology is given in Appendix A.

4.5.2 Standard Specification

The standard specification's 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, and is analogous to the use of `group by` in SQL. Specifically, it dictates which class represents a "row" or "record" within the faceted interface.

In this example, we have chosen the Piece facet as first-class, as all Pieces have an Album, and all Pieces have a Composer.

A sample of the data used in this example is given as RDF, serialised as N3 in Figure 4.8; the FacetOntology definition is given as RDF, serialised as N3 in Figure 4.7; and an illustration of the relationships is given in Figure 4.6.

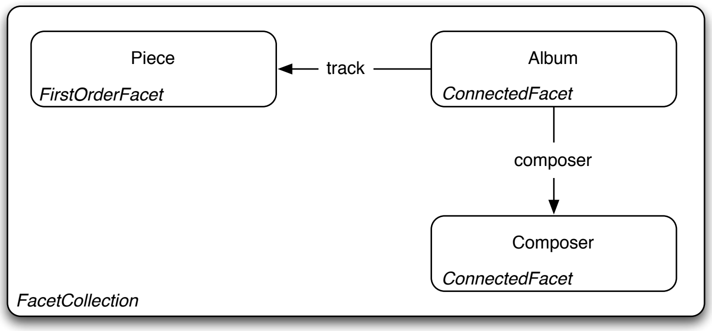


Figure 4.6: Example model of a StandardFacetCollection, illustrating connections between the Piece, Album and Composer classes, which shows the use of FirstOrderFacet and ConnectedFacet classes.

To define the Piece facet, we require only to define it using its RDFS Class, see Figure 4.7. 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`, see Figure 4.8. 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. 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://facetontology.example.com/mspace.n3#> .
@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#> .
@prefix d: <http://facetontology.example.com/data#> .

:mspace a facet:StandardFacetCollection .
:mspace facet:faceturi :item .
:mspace facet:rdfsource "http://facetontology.example.com/data.n3" .

:mspace facet:faceturi :piece .
:mspace facet:faceturi :album .
:mspace facet:faceturi :composer .

:piece a facet:FirstOrderFacet .
:piece a facet:Facet .
:piece facet:class d:Piece .
:piece rdfs:label "Piece" .

:album a facet:ConnectedFacet .
:album a facet:Facet .
:album facet:class d:Album .
:album rdfs:label "Album" .
:album facet:nextpredicate :album_predicate .

:album_predicate a facet:Predicate .
:album_predicate facet:predicateuri d:track .
:album_predicate facet:reverse "True"^^xsd:boolean .

:composer a facet:ConnectedFacet .
:composer a facet:Facet .
:composer facet:class d:Composer .
:composer rdfs:label "Composer" .
:composer facet:nextpredicate :composer_predicate .

:composer_predicate a facet:Predicate .
:composer_predicate facet:predicateuri d:track .
:composer_predicate facet:reverse "True"^^xsd:boolean .
:composer_predicate facet:nextpredicate :composer_predicate2 .

:composer_predicate2 a facet:Predicate .
:composer_predicate2 facet:predicateuri d:composer .

# nb: facet:reverse is not defined for composer_predicate2
```

FIGURE 4.7: Example N3 definition of a FacetOntology for the running example, as illustrated in Figure 4.6.

In addition to the above simple example, there is another case where data can be modelled that requires an additional definition. Specifically when only literals from the

```
@prefix : <http://facetontology.example.com/data#> .
@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#> .
@prefix d: <http://facetontology.example.com/data#> .

d:album1 a d:Album .
d:album1 d:track d:piece1 .
d:album1 d:track d:piece2 .
d:album1 d:track d:piece3 .
d:album1 d:composer d:composer1 .
d:album1 rdfs:label "Best of Bach" .

d:piece1 a d:Piece .
d:piece1 rdfs:label "Cantata BWV 1" .
d:piece1 d:originalName "Wie sch\"{o}n leuchtet der Morgenstern" .

d:piece2 a d:Piece .
d:piece2 rdfs:label "Cantata BWV 2" .
d:piece2 d:originalName "Ach Gott, vom Himmel sieh darein" .

d:piece3 a d:Piece .
d:piece3 rdfs:label "Cantata BWV 3" .
d:piece3 d:originalName "Ach Gott, wie manches Herzeleid" .

d:composer1 a d:Composer .
d:composer1 rdfs:label "Johann Sebastian Bach" .
```

Figure 4.8: Example N3 definition of the data used by the running example, as illustrated in Figure 4.6.

FirstOrderFacet are defined. In our example the user may wish to have a facet for the "Original Name," as specified by the predicate `d:originalName` (see Figure 4.8). In this example, if we use the `nextpredicate` definition by specifying `d:originalName`, the query engine will try to find an instance at that predicate, which will fail since there is only a literal. Thus, instead we specify that this Facet has a Facet Type (`facet:facettype`), and that it is `facet:TypeLiteral`. We then add a definition that is also has a label URI (`facet:labeluri`) of `d:originalName`. This informs the engine that the facet instances are not RDF instances, and are instead string literals. This also means that internally, instance URIs are not used to uniquely identify the instances, their string values are used instead. See Figure 4.9 for an example definition of the `originalName` Facet.

4.5.3 Advanced Specification

The FacetOntology specification includes additional directives that enable transformations to be processed against the input data before it is used in the faceted interface. For example by specifying a maximum value of a numeric facet, or to extract the year from a date field.

Specifically, Table 4.1 outlines the predicate directives that can be used against a FacetCollection, and Table 4.2 outlines the directives that can be used against Facets.

By using the mSpace Maker specific extension to FacetOntology (`facet:makertype`

```
@prefix : <http://facetontology.example.com/mspace.n3#> .
@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#> .
@prefix d: <http://facetontology.example.com/data#> .

:mspace facet:faceturi :originalname .

:originalname a facet:ConnectedFacet .
:originalname a facet:Facet .
:originalname facet:class d:Piece .
:originalname rdfs:label "Original Name" .
:originalname facet:type facet:TypeLiteral .
:originalname facet:labeluri d:originalName .
```

Figure 4.9: Example N3 definition of a Facet for "Original Name," demonstrating the use of the `TypeLiteral` property, using FacetOntology for the running example.

Directive	Value	Description
`facet:facetindex`	*non-null*	Include this facet in the keyword indexes
`facet:makertype`	`‘‘normal’’`	mSpace Maker specific setting to bound facets only by first order facet.
`facet:makertype`	`‘‘new’’`	mSpace Maker specific setting to bound facets by subpredicates, so that relational integrity from the RDF is maintained within mSpace.
`facet:datasource`	*literal*	The URL of a SPARQL endpoint to use as a data source.
`facet:rdfsource`	*literal*	The URI of a RDF N3/Turtle file to use as a data source.
`facet:rdfxmlsource`	*literal*	The URI of a RDF/XML file to use as a data source.
`facet:site`	*literal*	mSpace Maker specific setting to specify the name of the template folder to use when making the mSpace. Thus controlling the HTML and CSS style of the resulting interface.

Table 4.1: FacetOntology directives that can be applied to a facet collection.

Directive	Value	Description
`facet:preprocess`	iso8601:striptime	Remove the time from an ISO8601 formatted date.
`facet:preprocess`	iso8601:year	Extract the year from an ISO8601 formatted date.
`facet:maxval`	*integer*	Skip data that has a value greater than specified.
`facet:type`	`facet:TypeLiteral`	e.g. facet:TypeLiteral
`facet:type`	`facet:TypeObject`	e.g. facet:TypeLiteral
`facet:labeluri`	*predicate uri*	e.g. dc:date
`facet:class`	*class uri*	e.g. d:Item
`facet:sort`	`facet:SortNumeric`	Numerically sort this facet
`rdfs:label`	*literal*	The label of the facet
`facet:columngroup`	`literal`	
`rdf:type`	`facet:FirstOrderFacet`	This facet is the first order facet and the other facets are grouped by instances of this facet
`rdf:type`	`facet:ConnectedFacet`	This facet is a facet of instances of the first order facet
`facet:excludefromslice`	`‘‘true’’`	If present, this facet should not be shown in the slice, and should only be used in the result summaries. This is used for metadata that is inappropriate for use in columns, such a long descriptions.
`facet:truncatelabel`	*integer*	If present, the label of this facet is truncated to the number of words specified.
`facet:generate_summaries`	*non-null*	If present, a summary should be generated by the faceted interface for this facet, when displaying a keyword search result (e.g. to highlight the matched words).
`facet:summary_around`	*integer*	If present, the summary is truncate to this number of words “around” the matched phrase. Specifically, this means that the summary will include the matched phrase, and that it will contain this number of words before and after the match.

Table 4.2: FacetOntology directives for transformation of data, applied to each facet.

set to `new`, see Table 4.1), the graph integrity across predicates is retained through inter-column relationships in mSpace. An example of where inter-column relationships are important is when multiple columns represent items that are hierarchically related, and relationships between them are many-to-one, one-to-many or many-to-many. For example, consider a domain which holds hierarchical geographic location data of places mentioned in books, and those books may have multiple places mentioned in them, using the dimensions *Country* and *City*. One book may have two places mentioned, *London* in the *UK* and *New York* in the *USA*. Without enforcing a link between these columns, when a user selects *London* in the city column, the country column will show *UK* and *USA*, which falsely infers that *London* is in both the *UK* and the *USA*. This is because the query finds all books that have the city *London*, and queries for all of the *Countries* that those books have listed. A much clearer interface would enforce the hierarchical relationship between these two columns and show only *UK*.

In general, the *advanced specification* is valueable as it is generic enough that it allows any schema of data to be described fully.

4.5.4 Inferring Classes for use in Defining Facets

One of the key parts of facet definition is specifying the class to which the facets must be types. It may not always be the case that the data one wishes to appear in the facet already conforms to a particular class. In this case, the solution is to use inference to map knowledge into a class. This class could be one which already exists (although, if it did, the knowledge would likely already be typed as that class, or isn't typed to that class for a reason), or a new class defined for the purpose of this facet. One of the benefits to this approach is that the facet definition then retains a description that utilises a single RDFS/OWL class URI, which is therefore easily mappable by facets in other domains, in order to take advantage of using pivoting. Furthermore, inference can also be used as a "shim" to force crossover to occur, by superclassing the class used by a facet in another domain you wish to pivot with, so that your knowledge maps directly into another facet's definition, for the purposes of creating a faceted browser. In order to generate new information conforming to the inferred types, a reasoner is used, and the new inferred information is then asserted into the knowledge base with the original information.

4.6 Using mSpace Maker to Enable Users to Create Faceted Browsers

In order to demonstrate and prototype our architecture, we created a support tool called the mSpace Maker, which automates the creation of pivot-aware mSpaces, gathers data in the mSpace Server, and populates indexes in Facet Portal, based off a Facet Ontology definition.

Briefly, the steps followed by the mSpace Maker are:

- Read FO definition.
- Gather RDF that is specified in the FO as the source data, and assert in to a triple store temporarily.
- Using the FO definitions of the facets, query the combined RDF for the data.
- Copy the data into a relational database suitable for use in mSpace.
- Index the URIs of the data, and send, along with the FO definition, to a Facet Portal.

Upon completion of these steps, a URL to an mSpace is returned to the user, and the facet portal is now aware of the data and that the interface exists. This means that every other mSpace that uses the same Facet Portal (in this case, a single FP is used for everything made by the mSpace Maker) will return pivot opportunities to the user based on the new data, for any other mSpaces that have overlapping facets. While we have used mSpace for this prototype, this approach is not specific to mSpace, and could be applied to any faceted UI.

These components are shown in Figure 4.10, which illustrates the data flow through the components. A Facet Ontology description provides metadata which describes how data on the Semantic Web should be queried in order to be (automatically) gathered for use in a faceted interface. This facet ontology description is submitted to the mSpace Maker, which loads the definition, and gathers data from sources specified in the definition, supporting URIs of both SPARQL end-points and individual RDF documents. The sourced data is dynamically aggregated in a triplestore as a new named graph, and queried with SPARQL using the definition of each individual facet in the submitted description. A new pivot-aware mSpace interface is then created, and the details of its facets and the URIs of the data it contains are passed to the Facet Portal. By updating the facet portal, it ensures that all pivot-aware mSpaces that query the facet portal using its web services will be able to support our pivoting approach for any facets that are shared with the newly created mSpace. We recommend the use of this architecture, however the technologies do not prescribe it.

4.6.1 Performance of Automatically Created mSpaces

In order to deliver optimal performance we took our performance optimisations from the mSpace Model (see Section 4.4), and improved them further.

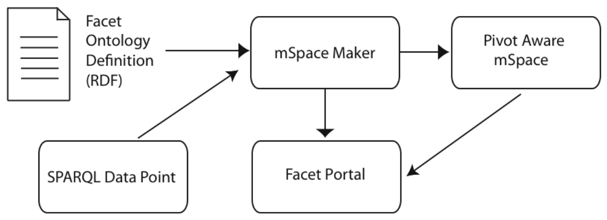


FIGURE 4.10: The architecture of our prototype is illustrated, showing the path of the data flow through the individual components, from facet ontology definition and Semantic Web data, through the mSpace Maker out to a pivot-aware mSpace that is indexed by the Facet Portal.

Specifically, it is important that the mSpace Maker incorporated all known performance optimisations as it creates the interface, because the created interfaces will have no further maintenance or tweaking performed on them. However, because we are creating the databases for the interfaces from scratch, we are free from any constraints that might ordinarily be the case when inheriting a legacy system.

High levels of query performance in a database can be achieved by:

1. **Reducing the total size of the database in memory** By eliminating redundancy and using the more efficient data types in each field. Specifically the system analyses the range of values used in each field, and uses the smallest possible representation format for the fields. Any literals are encoded using `varchar`s, with a maximum length specified as the maximum literal for that field in a specific table. Likewise all indexes created using a field are length limited to the maximum length of the values in that field.

2. **Lowering the memory required for table joins** By reducing the size of tables that are joined during queries. In order to reduce the size of tables, we create separate tables that represent the literals for each property (see Figure 4.11). Thus, each table has two columns, either two ints (which represent a relationship between two nodes) or an int and a varchar, which represents the link between a node and a literal.

3. **Minimising the number of rows evaluated during queries** By ensuring that all queries from the interface correspond to indexes. Thus, all tables have two compound indexes representing both directions of the relationship, for example a table containing the fields `myid` and `parentid` has the indexes (`myid`, `parentid`) and (`parentid`, `myid`).

4. **Eliminating paging to disk** By ensuring that queries perform all operations in memory. This is achieved by reducing the sizes of tables (as described above), and also by increasing the settings in MySQL to enable it to use as much memory as is available. By using the InnoDB storage engine, as opposed to the default MyISAM

storage engine, we have more control over the amount of memory we can use, above the limits of MyISAM.

4.6.2 Optimising Graph Structures in Databases: A Worked Example

In this subsection we work through a small example of the database tables that optimally represent a subset of a person's definition using FOAF. Specifically a person that has two data properties: `family_name` and `name`, and a single object property: `nearestAirport`, which has two data properties: `iataCode` and `title`. The table structure is illustrated in Figure 4.11. A listing of all of the fields and what they represent is given in Table 4.3.

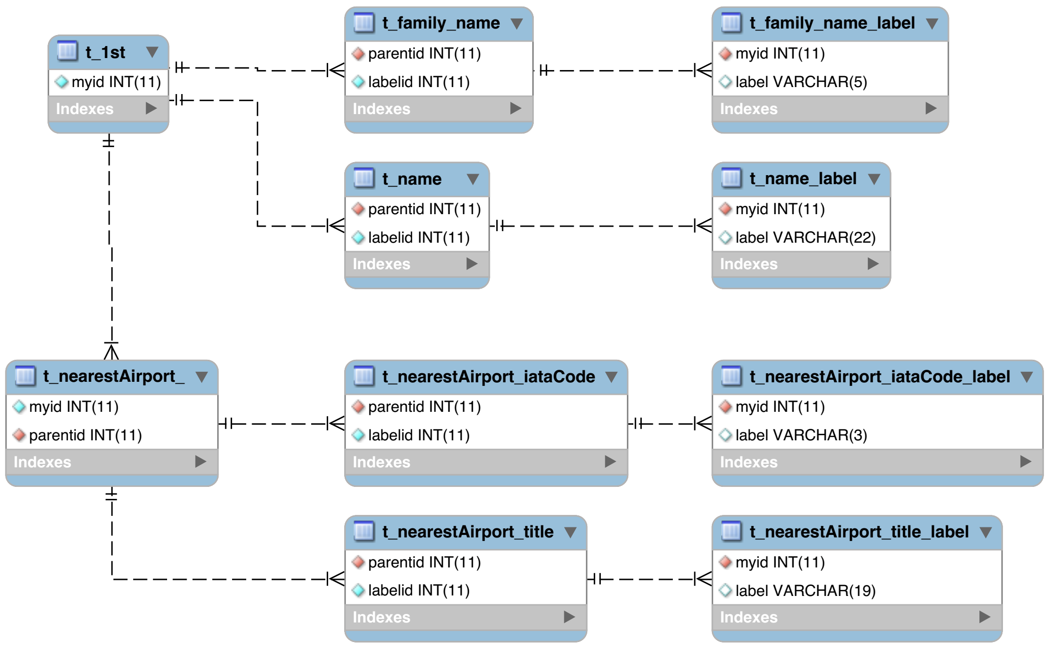


FIGURE 4.11: An example class diagram of a the tables in a database created by the mSpace Maker, which illustrates the separation of string data from the underlying graph structure, and the retention of the hierarchical relationships.

In general, by employing the above methods, we ensure that all queries will be satisfied using indexes and in-memory joins with the minimum amount of memory. The benefits are that queries will return much faster, because the query satisfier does less work. Consequently, the system can support more concurrent users, because queries return quickly, and their memory footprint is much lower.

4.7 The Data Picking Interface: User-created Interfaces

Our research focuses on presenting users with a lightweight interface that allows them to see the data available from a Semantic Web source, and focuses on highlighting the data

Table	Field	Content
t_1st	*myid*	Primary keys of all of the first order items in the domain. This field is used to join all of the other fields by the first order items.
t_family_name	*parentid*	This table represents the family_name data property. This field is a foreign key of t_1st.*myid* which links this predicate to a first order item.
t_family_name	*labelid*	This field is a foreign key of t_family_name_label.*myid* which links this table to a table of the human readable literals of this predicate.
t_family_name_label	*myid*	This table holds the unique labels for the family_name predicate. This fields holds an ID of the label.
t_family_name_label	*label*	This field holds the string label for this predicate.
t_name	*parentid*	This table represents the name data property. This field is a foreign key of t_1st.*myid* which links this predicate to a first order item.
t_name	*labelid*	This field is a foreign key of t_name_label.*myid* which links this table to a table of the human readable literals of this predicate.
t_name_label	*myid*	This table holds the unique labels for the name predicate. This fields holds an ID of the label.
t_name_label	*label*	This field holds the string label for this predicate.
t_nearestAirport_	*myid*	This table represents the objects that connect to the first order items via the nearestAirport predicate. This field represents the URIs of the items (as an automatically incremented integer).
t_nearestAirport_	*parentid*	This field is a foreign key of t_1st.*myid* which links this predicate to a first order item.
t_nearestAirport_ iataCode	*parentid*	This table represents the iataCode data property. This field is a foreign key of t_nearestAirport.*myid* which links this predicate to an airport.
t_nearestAirport_ iataCode	*labelid*	This field is a foreign key of t_nearestAirport_iataCode_label.*myid* which links this table to a table of the human readable literals of this predicate.
t_nearestAirport_ iataCode_label	*myid*	his table holds the unique labels for the iataCode predicate. This fields holds an ID of the label.
t_nearestAirport_ iataCode_label	*label*	This field holds the string label for this predicate.
t_nearestAirport_ title	*parentid*	This table represents the title data property. This field is a foreign key of t_nearestAirport.*myid* which links this predicate to an airport.
t_nearestAirport_ title	*labelid*	This field is a foreign key of t_nearestAirport_title_label.*myid* which links this table to a table of the human readable literals of this predicate.
t_nearestAirport_ title_label	*myid*	his table holds the unique labels for the title predicate. This fields holds an ID of the label.
t_nearestAirport_ title_label	*label*	This field holds the string label for this predicate.

Table 4.3: A listing of all of the fields in the worked example, and what they represent.

available from that source, in order for users to create a faceted interface over that data. Our approach attempts to lower the barrier to entry of creating a faceted interface, by automating as much of the process as possible, and using sensible defaults.

Our interface provides the user with a lightweight interface where RDF sources can be discovered from multiple locations, selected as data input sources for a new interface, and attributed mapped into a new faceted browsing interface.

4.7.1 Architectural Overview

In order to support user creation of interfaces, we have designed an architecture that handles the following key abilities:

1. Data Processing: Loading data source files (RDF) from the web into local knowledge bases, ready for querying.
2. Data Picking: Interface for exploring knowledge bases (created by the Data Processing engine, above), allowing users to decide which facets of the data to include in their interface.

An overall architecture for the combined system is shown in Figure 4.12, illustrating the inputs and outputs to the system.

In the following sections we describe the Data Processing and Data Picking subsystems in more detail.

4.7.2 Data Processing Engine

In order to support the gathering of large scale RDF data, we have created a data processing engine to handle this part of the backend. A breakdown of the backend architecture of the Data Processing Engine is shown in Figure 4.13.

The data processing engine handles internal management of RDF sources in triplestore knowledge bases. The triplestore then allows the knowledge bases to be queried using SPARQL. The engine is necessary as it allows any front-end interfaces to data to be able to use SPARQL as a generic query interface regardless of the specifics of the serialisation or of the hosting of the data. For example, some data providers split their data amongst different categories, or as linked data, split up amongst many small files. By using a data processing engine, we abstract the task of web-based data collection to this purpose-built engine, and away from the user interface. Additionally, the design of the data processing engine is generic, so that if we need to swap out the backend triplestore software with that of a different vendor, this is a simplified process that will only require writing a

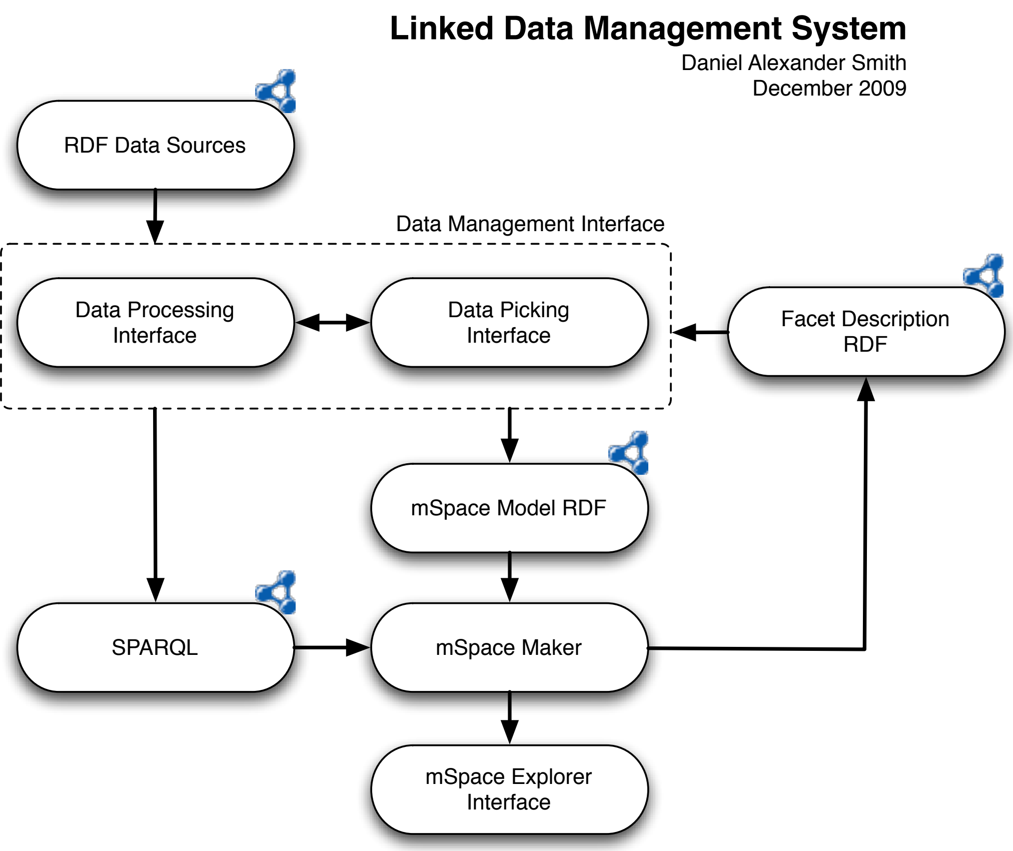


FIGURE 4.12: Architectural overview diagram of the backend system to support User Created Interfaces.

shim interface that conforms to our data processing engine's API. For these experiments we are using OpenLink Virtuoso[7].

4.7.3 Data Picker Interface

In order to allow users to explore data sources and create interfaces from them, we have designed a prototype *data picker interface*. The interface allows users to load data sources (using the Data Processing engine described in Section 4.7.2 behind the scenes), and explore them for facets of interest. A breakdown of the backend architecture of the Data Picking Interface is shown in Figure 4.14.

Due to the potential complexity of the task of exploring large data sets, there is potential for an interface to become overwhelming if not carefully designed to prevent this. As such, when designing the interface for the data picker, we have been mindful to attempt to make the interface as lightweight as possible.

One of the ways in which we have done this, is to design the interface to default-include

[7]OpenLink Virtuoso: `http://virtuoso.openlinksw.com/`

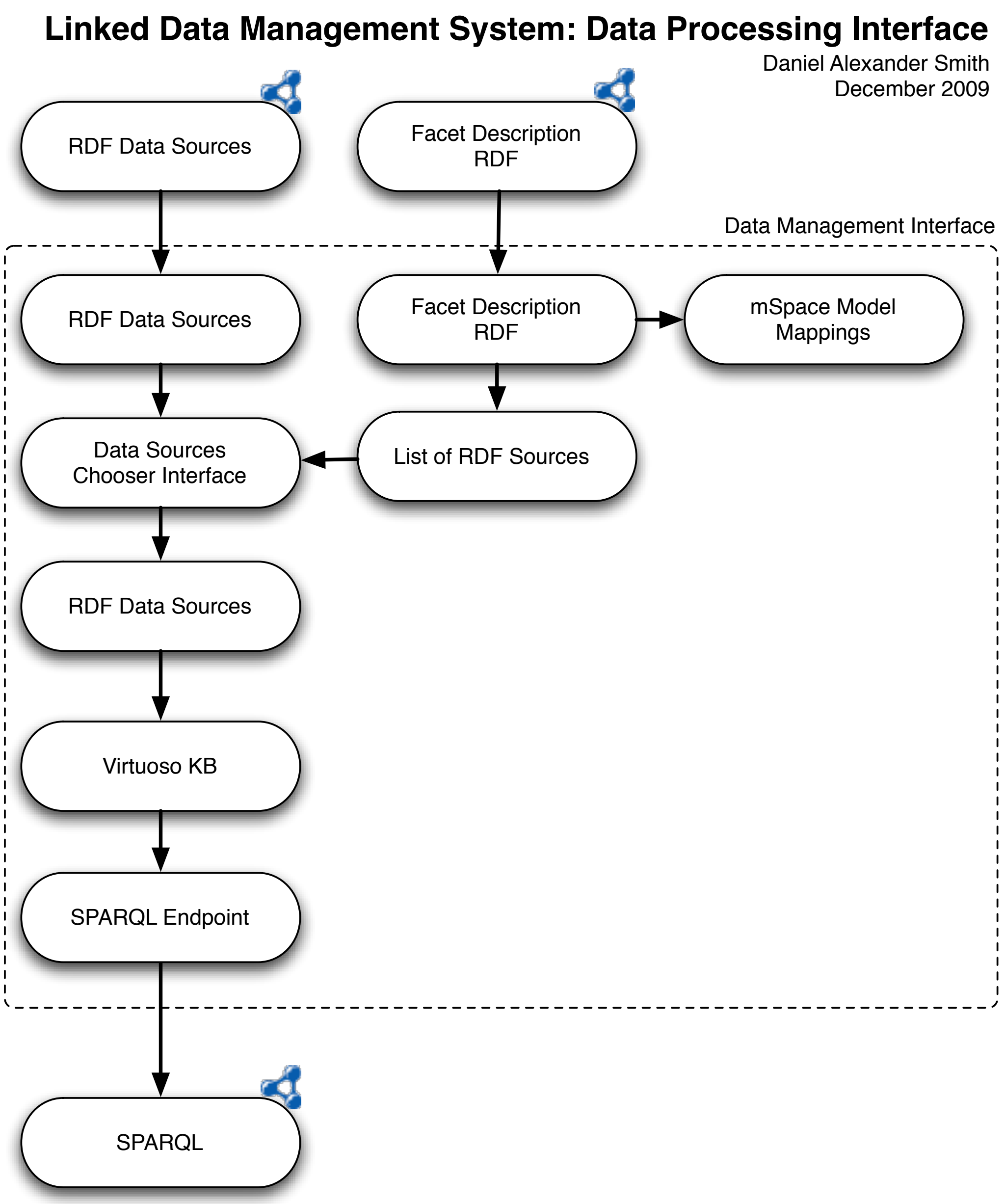


FIGURE 4.13: Architectural overview diagram of the data processing interface to support User Created Interfaces.

data, rather than require the user to specify that the data they are exploring should be shown in the final interface that they are creating.

Through the data picker interface it is possible to create an interface in very few steps. The shortest set of interactions is as follows:

1. Click, or enter the URI of a SPARQL endpoint, and click submit.
2. Select a Class from the list.
3. Click "make mSpace."

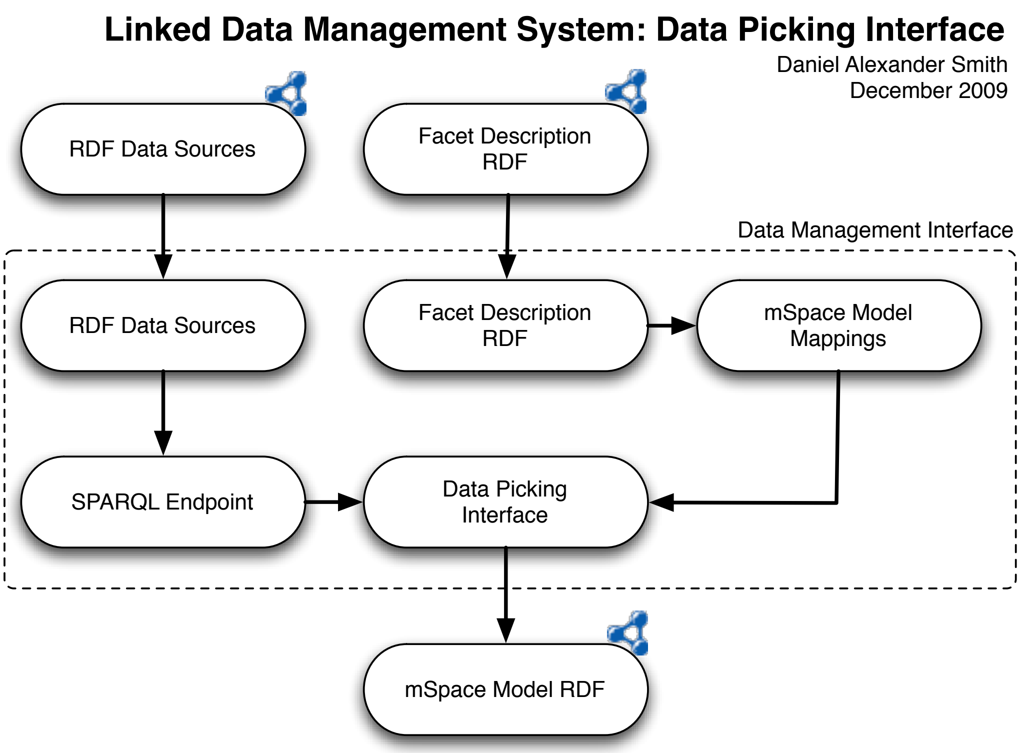


Figure 4.14: Architectural overview diagram of the data picking interface to support User Created Interfaces.

By following these actions, the user will be presented with an mSpace interface that contains data from the chosen SPARQL endpoint. A typical interaction with the system is illustrated in a sequence diagram in Figure 4.15.

A user loads the Data Picker (Fig. 4.15, **A**) and is presented with the interface, which allows them to select their data source (Fig. 4.15, **B**). The user can either select a SPARQL endpoint from a list, type one in (if they know the URL of an endpoint), or then can add an RDF file by its URI. If they add an RDF file, the data picker needs to add it to a local knowledge base before it can be queried. In order to do this, it invokes (Fig. 4.15, **C**) the Data Processor (see Section 4.7.2), which collects the knowledge from the Semantic Web (Fig. 4.15, **D, E**), and into a local knowledge base (Fig. 4.15, **F**). The Data Picker then queries the knowledge base for all RDF(S)/OWL classes and displays these to the user (Fig. 4.15, **G**). The user picks the class that represents the type of record that they wish to browse over (Fig. 4.15, **H**). The Data Picker then queries all objects of this type, determines which predicates link or from those objects. These predicates are then shown to the user (Fig. 4.15, **I**), and used a basis for creating facets. A sample interface is then shown to the user, and they have the option of altering these facets if they wish (Fig. 4.15, **J**), before creating the final faceted interface over the data (Fig. 4.15, **K**).

In the next section we describe in more detail the actions that the user can take when

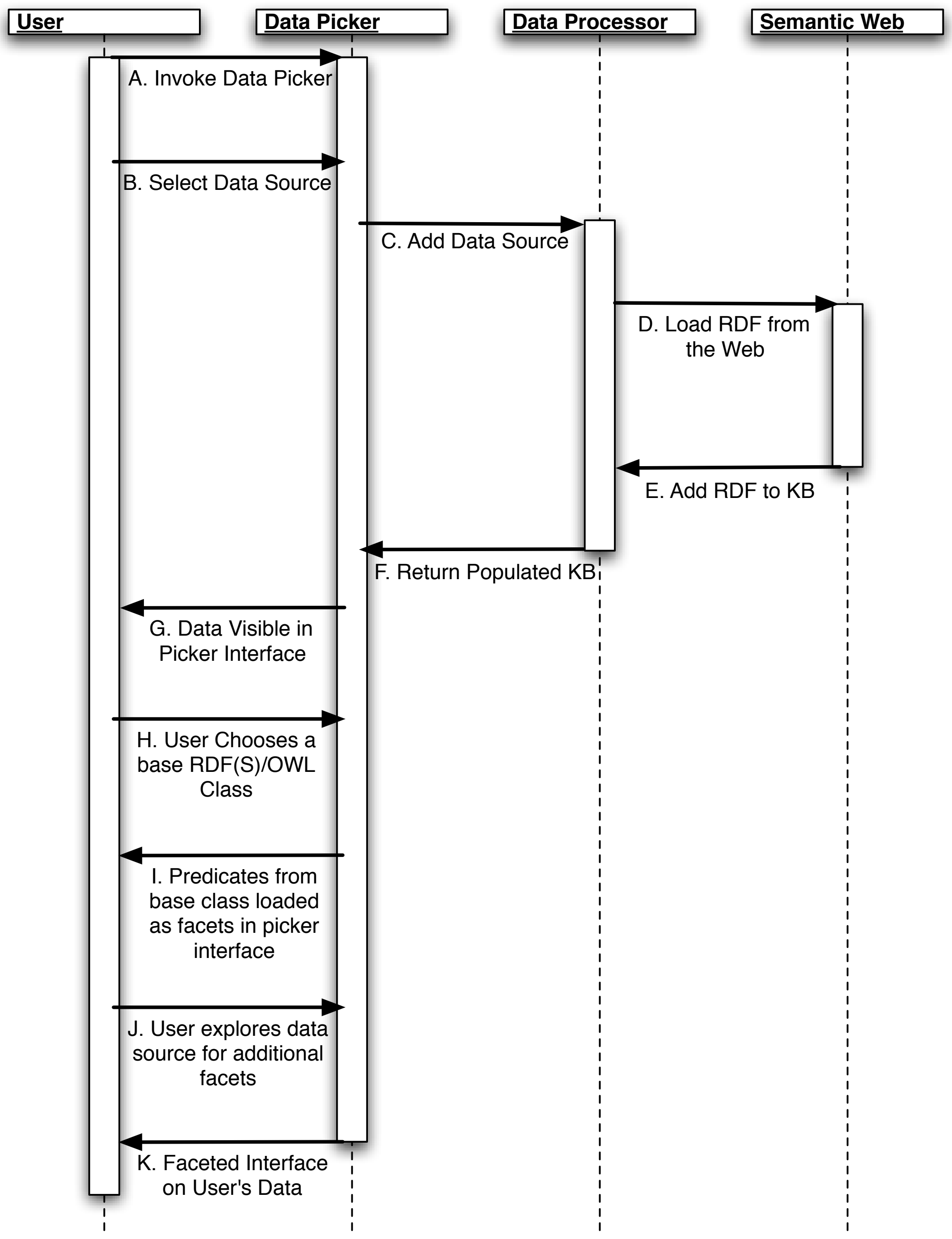

User
Data Picker
Data Processor
Semantic Web
A. Invoke Data Picker
B. Select Data Source
C. Add Data Source
D. Load RDF from the Web
E. Add RDF to KB
F. Return Populated KB
G. Data Visible in Picker Interface
H. User Chooses a base RDF(S)/OWL Class
I. Predicates from base class loaded as facets in picker interface
J. User explores data source for additional facets
K. Faceted Interface on User's Data

FIGURE 4.15: Sequence diagram showing how a user interacts with the data picker, how the data picker interacts with data processor, and the data processor interacting with the Semantic Web.

creating the faceted interface, and the benefits to the user.

4.7.4 In Use

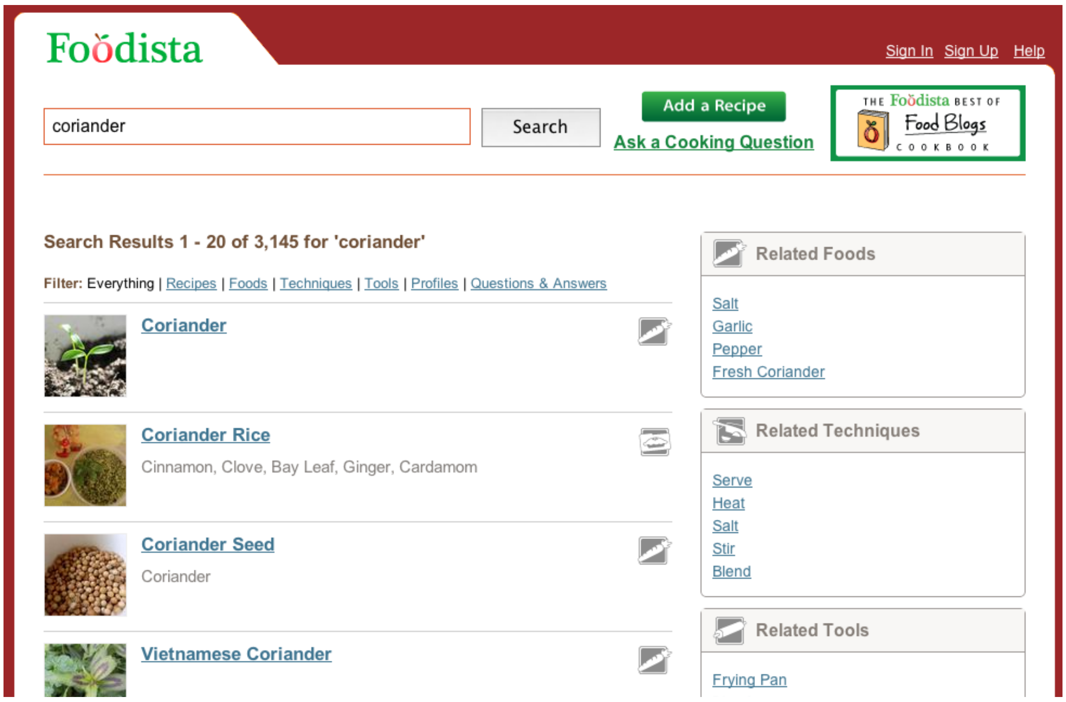
Through the use of the Data Picker, a Facet Ontology definition of the data source is created. This definition is used to create the mSpace faceted interface over the user's data. The availability of the Facet Ontology definition have benefits in itself, above the creation of the mSpace. Specifically, the definition can be loaded into the Data Picker by other users that wish to create a view on the same data source, but customise which facets they want to view, and build upon the work of the first user. In our test cases, large scale data sources were used, which leveraged the scalability of the mSpace Maker (as described in Section 4.6), to provide large-scale exploration. Facet Ontology is described in more detail in Section 4.5.

The user benefits by having a scalable faceted browser over their data. The same benefits of Exhibit apply here, in that publishers are able to allow their audience to experience and explore their data in additional ways, while being able to do this at a much larger scale than a client-side tool like Exhibit allows.

For example, a user may wish to query food recipe data from the web site *Foodista*[8]. The web site provides an highly structured wiki-like user editable recipe system that focuses on the concepts of *recipes*, *tools*, *foods* and *techniques*. Thus, a recipe links to the foods it comprises, and the tools and techniques required to prepare it. The website also offers the ability to browse from tools, foods and techniques back to recipes. Already the site provides a reasonably powerful interface, however, for more complex queries than single items, the user has to manually collate results. For example a user may wish to try out a new blender they have received as a gift, and is looking for recipes that include carrot and coriander that use a blender. While this information is likely on the site, the user would have to choose to browse by looking at all blender recipes, all carrot recipes or all coriander recipes (see Figure 4.16), and examine them all in order to determine which match their requirements. However, through the use of a faceted browser, they could instead simply select "blender" from the "tools" facet, and "carrot" and "coriander" from the "foods" facet, resulting in recipes related to all of these items being listed.

In the next section we will discuss how we can pivot across domains with interfaces that have been defined using the Facet Ontology.

[8]Foodista: `http://www.foodista.com/`


Foodista

Sign In Sign Up Help

coriander Search

Add a Recipe

Ask a Cooking Question

THE Foodista BEST OF Food Blogs COOKBOOK

Search Results 1 - 20 of 3,145 for 'coriander'

Filter: Everything | Recipes | Foods | Techniques | Tools | Profiles | Questions & Answers

Coriander

Coriander Rice
Cinnamon, Clove, Bay Leaf, Ginger, Cardamom

Coriander Seed
Coriander

Vietnamese Coriander

Related Foods
Salt
Garlic
Pepper
Fresh Coriander

Related Techniques
Serve
Heat
Salt
Stir
Blend

Related Tools
Frying Pan

FIGURE 4.16: Screenshot of the Foodista website, showing the search results for Coriander. There is no possible way to search for recipes that contain two named ingredients, and thus the user has to manually collate recipes in order to fulfill such a need.

4.8 Semantic Pivoting Across Heterogeneous Domains

In this section we focus on the challenge of enabling users to explore related data across sites and domains. This is challenging because different domains mark up their data differently, and do not provide any mechanisms to merge or explore their data along with data from external sites. For example, to allow users to browse from musical artist information on one site, to information about events on another site, alongside information about the price of CDs from various online music stores.

In order to address this challenge we build on our existing work: In the last chapter, we addressed the problem of enabling users to create their own faceted interfaces over large scale structured data sources by contributing the *FacetOntology* ontology. This ontology defines an abstraction over RDF data for use by faceted interfaces. We also addressed the problem of creating such an abstraction by contributing the *Data Picker* interface which enables users with no knowledge of the Semantic Web to select data patterns from Linked Data, create a FacetOntology model, and generate their own interface over the picked data.

While these approaches enable us to model mechanisms that make a single domain explorable, such a model does not allow any browsing outside of that individually defined domain. This is because this model would not have access to any information outside of a single FacetOntology model, and if it did, it has no way to link heterogeneous data from multiple sources. One way that linkage between heterogeneous data has been achieved is in the domain of organisational databases, through the use of "pivoting," (Robertson et al., 2002a). Specifically, Robertson et al. use visualisations of polyarchies are used to browse references to individual entities, such as an employees, when that reference is referenced in multiple databases. This worked well because those references were global across the databases, and because the databases were centrally maintained and limited to a single organisation. Our exploration is not limited to a single data source, and thus supports the exploration of heterogenous data. When an entity (such as an employee) is referenced in different sources using different identifiers, "co-reference ambiguity" occurs (Glaser et al., 2007). Thus, any exploratory interface that wishes to move across heterogeneous data sources needs to address this challenge.

In this chapter, we describe our approach that dynamically expands a domain of information that is used in an exploratory search interface, by enabling users to browse from information in one interface through to other related interfaces. We refer to this interaction as "pivoting," in the same way that (Robertson et al., 2002a) refers to pivoting from one database to another, our interaction pivots from one faceted area of information to another. In our approach, users can pivot from one interface to others when they share facets. For example, a user browsing an interface on the domain of food can pivot from a nutritional information dataset which contains information about the fat, salt and sugar content of foods, through to the price and availability of those items on an online grocery interface. The food item's name acts as the pivot because it is contained in both the nutritional information dataset and the online grocery interface. By identifying common facets in different interfaces, we can use those commonalities as pivots to enable users to query across domains, so that facets from any interface can be used as filters on all the other interfaces.

4.8.1 Semantic Pivoting

Semantic pivoting is a mechanism which allows users to combine metadata from multiple sources at the point of exploration, in order to filter information and explore using more facets than a single source offers. Previous work has shown that manually gathering metadata from multiple sources, and providing faceted exploration of the combined data set is an effective enhancement compared to browsing a single data source (m.c. schraefel et al., 2006). Semantic pivoting adds to this approach, by automatically determining if facets from other interfaces can be aligned with a data set, and presenting those facets to the user, so they can decide to add them to their faceted browser, in order to filter using the data in the added facets. When a new data set is marked up for faceted browsing, its facets are immediately available for inclusion into other already-defined faceted data sets. Thus, rather than having to wait for a specialist to create a new faceted browser that mashes up data from two faceted interfaces, a user can pick facets from multiple faceted data sets into a single browsing interface, using an interaction we refer to as *pivot-pull*. We hypothesise that by allowing a user to dynamically add metadata from multiple sources, based on their own needs, rather than those curated by a third party, we can further enhance a user's power to filter items based on specific search criteria using faceted browsing.

The World Wide Web contains a number of domain-specific sites that carry information about particular aspects of different items (where an item can be anything of specific interest to a web site's domain, and is typically a product, person or place), and due to licensing restrictions, technical limitations or other issues, there are often missing links between disparate data sources. For example, the web site *NutritionData.com* contains extremely detailed nutritional information on a large number of ingredients, food products and meals, but this information cannot be used to compare products while shopping online. Furthermore, while it is possible for a programmer to write a 'Greasemonkey' script[9] to add links to *NutritionData.com* onto shopping sites on a per-item basis, it does not enable sorting or filtering based on particular nutritional facets. For example, a user may require an increase in Folic Acid and a decrease in Saturated Fat in their diet, but there is no way to order the list of available items on their preferred online shopping website, *Ocado*[10], by their own chosen nutrient, even though that data is technically available online, at *NutritionData.com*[11]. It is prohibitively time-consuming for users to currently search for products on grocery shopping web sites by nutritional content, and we hypothesise that Semantic Pivoting will enable users to perform this type of action in a small amount of time, and using an intuitive interface. Thus, we have created a prototype using product data from *Ocado* and nutritional information from *NutritionData.com*, in order to address the challenge of pivoting across these domains.

At present, data collation for exploring heterogeneous data sources can indeed be performed manually by a trained human for any query: run a search of terms through a keyword search engine such as Google (Brin and Page, 1998), check and collate the results, assign results to subject headings (using domain knowledge), and iterate through the collated items looking for matches. This doesn't scale to the general case very well and on even trivial cases would require many *man hours*, indeed a research task like this, undertaken by "Knowledge Workers" (Drucker, 1974), is a common task and occupation. In order to expand this beyond a specific query of the user into the general case, and to incorporate as many sources as exist, doing this manually would limit you to few sources, due to time and resource constraints, making automated methods necessary.

The challenges in exploring heterogeneous data can be split into two distinct areas: that of connecting together heterogeneous data, and that of the exploration of the connected data.

[9]'Greasemonkey' is a browser extension that allows programmers to add information such as links in other people's websites to enhance the site in ways the web site publisher has not done.

[10]Ocado: `http://www.ocado.com/`

[11]NutritionData: `http://www.nutritiondata.com/`

The first problem with connecting sources is the analysis of each source itself. Without knowing what a source holds, one cannot connect it to other sources. It is therefore critical that determination of the domain is relatively accurate before any attempt is made at connecting it to another source, especially when that source has been determined using the same logic. We have shown that gains can be made in the representation of data, for the purpose of faceted browsing, by having the domain experts model the data with the technologists (Smith et al., 2009).

There are numerous challenges in the exploration of combined data sources. It appears clear that how to determine dimensions within a space, and indeed crossing between sources within this space is clear cut, however the challenge occurs when determining the optimal points to make these crossovers.

The key question is how to enable exploration from a single domain model to multiple domains. This work concerns how to determine the points that multiple domains intersect, such that a near-seamless exploration from one domain to another can be achieved through a lightweight UI metaphor, such that the user can explore two separate domains as if they were connected as one large domain. The context of the intersection has to make sense however, since once the domain switch has occurred, the original exploration no longer makes sense to have an effect, i.e. filtering can only affect a specific modelling domain, and this does not transfer across domains.

In the following section we discuss our approach to semi-automate the discovery of patterns in data graphs, such that specific attributes can be displayed and used as constraints in order to explore a data set, or aggregated collection of data sets.

4.8.2 Our Semantic Pivoting Approach

We present an approach to exploring (linked) data that we have called "Semantic Pivoting." We note that we are not the first to use this term, it has been used previously to describe the notion of the use of ambiguity in songwriting, where a term is used that means two things, and one meaning is implied up to a point in the song, when a "semantic pivot" happens, and the user is guided to the other meaning of the term, requiring them to rethink the meaning of the whole story so far (Pere, 2009). We use the term in a similar, albeit less obviously artistic way to represent a mechanism to enable contextual browsing across heterogeneous sources on the Semantic Web.

Specifically, a Semantic Pivot is a point where two datasets have an overlap in their data. For example, there is an overlap in data from a supermarket, and data from a nutrition information website. The overlap is that they both describe different aspects of food products, and thus the Semantic Pivot is over these *food products.*

We proceed by detailing the architecture of our prototype implementation.

4.8.3 Dynamic Association of Related Data: an Architecture to Support Faceted Browsers

Our approach to browsing provides a vocabulary to describe RDF data, allowing that description to be used to create a faceted browser, as described in Section 4.5, and using an intelligent system that can detect points in data that overlap with other datasets, so that users can browse from one dataset to another seamlessly. To enable that interaction, we draw on the approaches of Tabulator, mSpace and /facet, which have shown to be effective, and augment it in order to achieve benefits in scalability, interaction performance and reduced amount of ahead-of-time markup of data. To present solutions to the challenges faced by the above related work we present Facet Portal, Facet Alignment Service and interactions for Pivot-aware Browsing.

First, we present Facet Portal, a repository of Facet Ontology definitions that provides a web service API for Faceted Browsers to query to discover other faceted collections that they can pivot to.

Second, we present a Facet Alignment Service, a system that analyses metadata from multiple heterogeneous sources, to determine overlap between facets, and to attempt to perform co-reference disambiguation between entities in overlapping facets from different sources.

Finally, we present interface interactions for Pivot-aware Faceted Browsing, whereby we identify two interactions that are possible are a result of modifying a browser to utilise the Facet Portal.

By using our pivoting approach, we enable interfaces to have knowledge of how to connect to data outside their primary domain. We present a prototype implementation using two interface paradigms we call *pivot linking*, and *pivot pull*. *Pivot linking* is the simplest use of the pivot, whereby a link is shown to the user to bring up the other data set in a new window, with the selected item from one interface pre-selected in the overlapping facets of the second interface. *Pivot Pull* is a more advanced paradigm where a pivot-aware browser can take facets from other Facet Ontology definitions, and dynamically show within the interface, in order to provide the benefits of retaining context (Wilson et al., 2009). We do not intend to prescribe a particular methodology of how semantic data pivots should be used, and future work will explore additional ways that such pivots can be utilised to provide a richer exploration of data for users. By opening up the pivot possibilities to the wild, the possible paths to explore become rich quite quickly. The challenges are how to make these associated data sets discoverable and then how to have them map appropriately.

The following subsections describes the architecture of the demonstration interface, which is comprised of the Facet Portal, Facet Alignment Service and a Pivot-aware Browser.

4.8.4 Facet Portal: Aiding Discovery of Data and Pivots

In this section we describe *facet portal*, a web service API that provides faceted interfaces with information on pivots.

In order for faceted interfaces to link to other interfaces via pivots, they must be aware of data which overlaps with their domain. Facet Portal supports this by providing faceted interfaces with two indicators of overlap with other interfaces: facets that overlap, and individual metadata items that overlap. Additionally, Facet Portal provides URLs to use to link to the other interfaces, for use when users select items that overlap with other interfaces.

In order to scalably answer queries about which URIs are contained in each faceted interface, facet portal indexes the data as follows:

1. A facet portal is loaded with the facet ontology definitions of two or more different interfaces.
2. When being loaded with a new facet ontology definition, the facet portal gathers all of the RDF metadata from the data sources specified in the definition.
3. The portal then indexes all of the URIs of the data, their types (i.e. their RDF classes) and the predicates used to gather them.

From these indices, the portal can answer queries from an interface about which other interfaces share facets (by comparing their RDF classes), and within these shared facets, which instances they share. In order to enable such queries, the portal has to be able to globally identify individual pieces of metadata. We describe the process enabling such identification in the following subsection.

4.8.5 Global Identification of Metadata

In the case that the instance in the facet is an individual with a URI, it is the URI that is compared to determine they are the same. This is not always the case, however, since it is likely that some facets may be made up of metadata attributes. For example, a facet definition may contain the facet "calories," which has a numerical value, such as "107" being the calorie content of a "KitKat Bar". Assuming that a data provider would model a numerical value as its own individual is unrealistic, and while it would be possible to do so, it is more likely that it would instead be modelled as an attribute of the subject (in this case, an item of food), and that the unit of measurement would be encoded in the ontology (for example, that a `nutrition:calories` predicate refers to the calorie content in `KJ`). As such, the URI of the subject is not unique to the calorie metadata

"107", as it is the URI of an item of food, the "KitKat Bar," and therefore cannot be used on its own to refer to the item in the calorie facet. To address this challenge we instead we follow the approach of using the metadata as an IFP (inverse functional property), by indexing the URI of the predicate together with the value of the item (in this case `nutition:calories` and "107"), as the global identifier of the instance. This approach is similar to that used by the semantic search engine Sindice (Tummarello et al., 2007), which offers lookup by IFP as well as URI.

4.8.6 Facet Portal API

In our prototype we enable facet pivot information to be looked up live via a web service, which is called by an extension of the mSpace faceted browser. The browser calls the web service each time the user clicks on an item, asking the facet portal if there are any other interfaces that share the clicked item. From the results returned, the mSpace generates links to other interfaces (other mSpaces), that the user can click to pivot across to. The operation of the pivot-aware mSpace is described in more detail in Section 4.8.8.

We have designed a facet portal system to provide pivot information via either a web service API, or via downloadable RDF linked data containing the information that backs up the web service (for use in applications that can't use the web service). The architecture of facet portal has been designed so that facet portal services can be run by administrators of interfaces, to enable them to choose which other interfaces that users can pivot across to. We also encourage interface designers to allow the user to choose their own facet portal URI, so that the interface remains dynamic and global, taking advantage of new interfaces and domains as they are created and loaded into facet portals. To enable the creation of community facet portals, our prototype of a facet portal import and exports the Facet Ontology RDF definitions of interfaces. This also reduces the reliance on a single centralised facet portal, in keeping with the Semantic Web data model that underpins this work.

4.8.7 Facet Alignment Service

Cross-domain exploration is a natural paradigm for exploring knowledge and serves two purposes, firstly as a user interface metaphor that empowers users to context-switch across domains during a faceted browsing session, and secondly as a lightweight data alignment mechanism that permits tractable domain alignment on the Semantic Web. This is made possible by taking advantage of that hypothesis that if two different domains contain a facet of the same (RDFS/OWL) class, then they are linked (by that class). This linkage is then shared with the user, enabling them to "pivot" from one domain to another.

However, heterogeneous sources may not always use the same ontologies, and thus the classes may be different. Furthermore, if sources do not use the same ontologies to describe the classes (or *types*) of their data, they are unlikely to use the same identifiers to refer to their entities. Thus, in order to browse across heterogenous data sources that provide overlapping data about related domains, the challenge of entity co-reference must be addressed. Entity co-reference is the problem that while two data source make reference to identical concepts, they may not use the same terminology or identifiers to refer to them, and thus operations that involve both sources, such as browsing with semantic pivots, are unable to make the correct links through the data. To overcome this problem, we present the Facet Alignment Service.

The Facet Alignment Service has been designed with the recognition of the fact that automated co-reference cannot guarantee perfect results, with both false-positive and false-negative results to contend with. Thus, the service is comprised of an automated alignment system, and a lightweight management interface to allow domain experts to make assertions, alterations and additions to the results of the automated alignment system.

Automated alignment of Semantic Web concepts has been provided with tools such as the Alignment API (Euzenat, 2004), which provides the ability to load two ontologies, and returns percentages of the likelihood that class names from the different ontologies should be aligned. The choice as to the threshold suitable for using the results automatically is up the user of the alignment API. In our case the alignment weight is used to rank the automated alignments in the management interface, so that the domain experts that are confirming matches can check the highest weighted matches first, and assert that after a certain threshold enough matches are incorrect and all should therefore not be used. This design reinforces a regimented workflow for domain experts when carrying out the alignment overview task, whereby once they reach a point where alignment suggestions are no longer meaningful, they can stop working, as opposed to an alphabetised system (for example), where the workflow would be more akin to "wading through" a large number of incorrect low-weighted alignments.

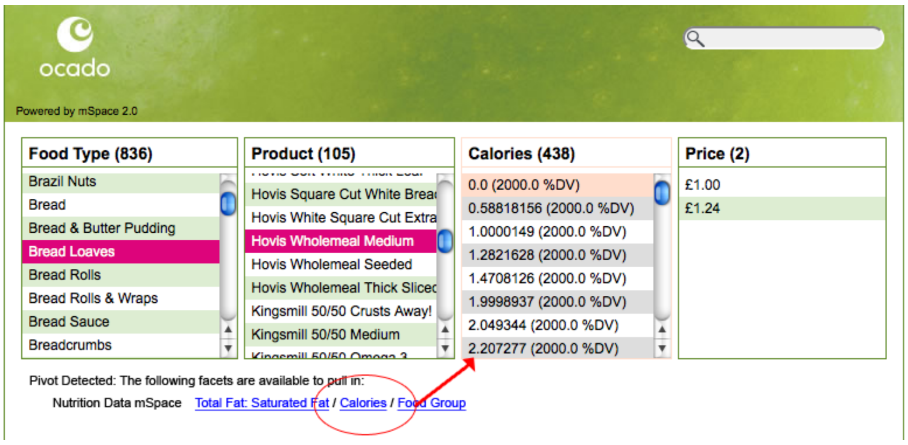
In the next section we describe how browser enhancements to enable pivot browsing to help users explore heterogeneous domain information.

4.8.8 Pivot-aware Browsing

In order for a user to browse from one faceted data set to another using pivots, a pivot-aware browser must be used. In our prototype we have used a modified version of the mSpace faceted browsing framework in order to demonstrate possible interaction methods for enabling semantic pivoting (while we have modified mSpace specifically, it is possible to modify any faceted browser to add pivoting support by interacting with the

Facet Portal API as described in Section 4.8.6). These alterations enabled the following two interactions:

1. When the mSpace first loads, a button is placed below each facet to enable facet-based semantic pivoting. When the button is clicked, a list of facets available on other interfaces is displayed to the user (see Figure 4.17). Each facet name displayed is a clickable link that opens a new facet column in the current mSpace. In our example we show a supermarket faceted interface, which provides links to facets from a Nutrition Data mSpace. The user has selected "Calories" (as indicated by the red overlay), and this facet has been *pivot pulled* into the supermarket browser. The user can continue to pull in additional facets as required.



Food Type (836)	Product (105)	Calories (438)	Price (2)
Brazil Nuts	Hovis Square Cut White Brea	0.0 (2000.0 %DV)	£1.00
Bread	Hovis White Square Cut Extra	0.58818156 (2000.0 %DV)	£1.24
Bread & Butter Pudding	Hovis Wholemeal Medium	1.0000149 (2000.0 %DV)	
Bread Loaves	Hovis Wholemeal Seeded	1.2821628 (2000.0 %DV)	
Bread Rolls	Hovis Wholemeal Thick Slicec	1.4708126 (2000.0 %DV)	
Bread Rolls & Wraps	Kingsmill 50/50 Crusts Away!	1.9998937 (2000.0 %DV)	
Bread Sauce	Kingsmill 50/50 Medium	2.049344 (2000.0 %DV)	
Breadcrumbs		2.207277 (2000.0 %DV)	

Pivot Detected: The following facets are available to pull in:
Nutrition Data mSpace Total Fat: Saturated Fat / Calories / Food Group

FIGURE 4.17: Screenshot of the pivot-aware mSpace interface showing a list of facets that can be pivot-pulled from the Nutrition mSpace, with "Calories" pulled in.

2. When the user clicks an item in one of the mSpace columns, information is shown below the column set. In the pivot-aware mSpace, this information is now preceded by a list of instances in other mSpaces that are equivalent to the selected instance. Each listed instance is a clickable link that opens the pivotable mSpace in a new window, with the pivoted instance pre-selected in the newly opened mSpace. We call this interaction *pivot linking*. All of the columns displayed in the new window are filtered by the selected instance, ensuring that the interface only shows metadata related to the pivoted item. This interaction takes a typical web approach to linking, and maintains the faceted browser that is used by a particular dataset, and is useful if two different faceted browser frameworks are used, but still wish to support our pivoting approach at the lowest level.

One of the important considerations when designing this API is that of allowing multiple selection of items to be supported. Specifically this is because the API should be able to support scenarios where not only does a user wish to pivot over a single item, but the entire contents of a filted facet column. For example, consider a case in the supermarket/nutrition semantic pivoting example data sets. A user wishes to know the prices of all foods that provide less than 0.5% saturated fat, but over 300 calories per 100mg. In order to do this, the user would start with the nutrition interface, select an upper bound of 0.5% in the saturated fat column (effectively selecting everything from 0 to 0.5%), and a lower bound of 300 calories. Following these selections, the food type column is populated with items that match these filters. In order to find out the price of all of these items, the user must pivot on all of the items in this column, and select them all in the new interface. Our pivot-aware mSpace supports this interaction using the *pivot pull* technique shown above, where the user is not made aware of the back-end API interactions.

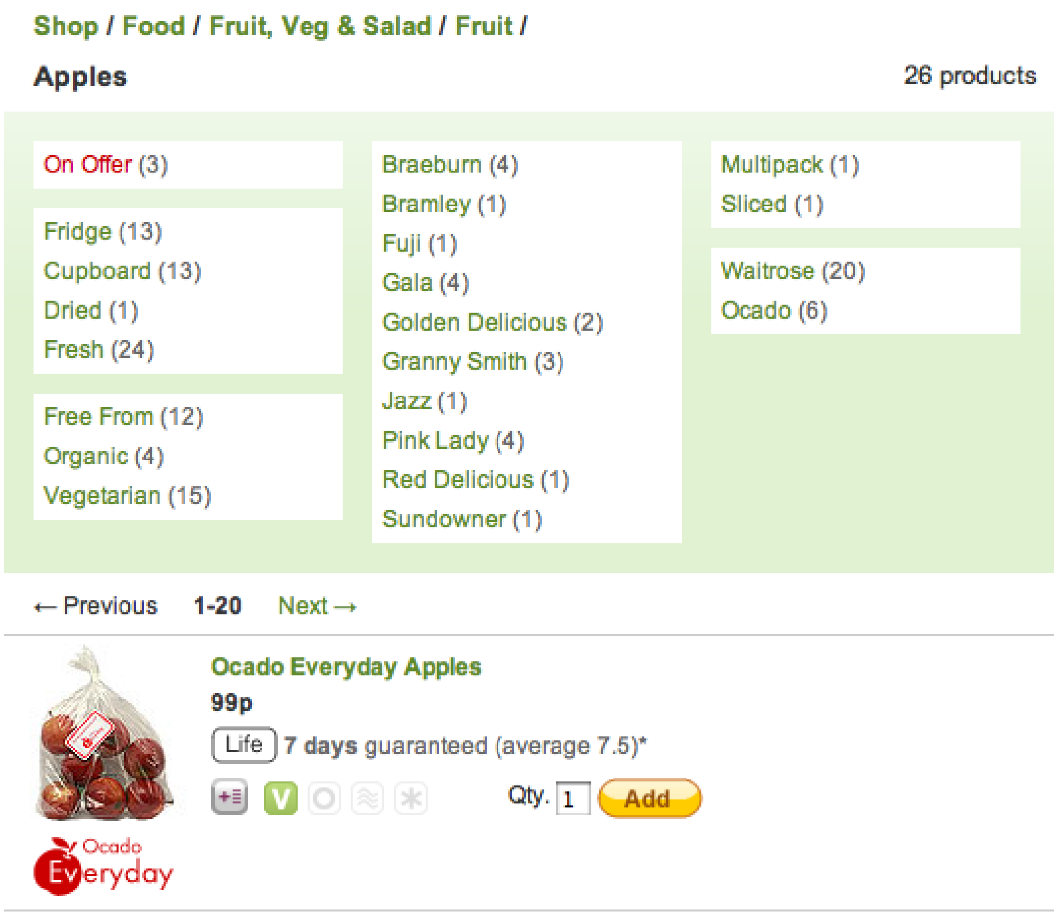
These interactions enable the mSpace to link automatically to other interfaces that contain pivots. These links to other interfaces are generated by a web service, and not by an individual interface. This means that they can be updated and changed dynamically without making changes to the interface. This allows links to new interfaces to be displayed in all relevant existing interfaces, and also the bi-directionality of links are preserved, therefore it will always be possible to go back to the source interface, allowing users to re-trace their interaction choices. The ability to retrace their choices is important, as the recoverability of context has been shown to be beneficial to cognitive understanding during exploration (Wilson et al., 2009).

In the next section we walk through an application of our approach on a sample dataset, in order to address the challenge of building knowledge over two domains related to food and health.

4.8.9 Pivoting Across the Food Domain — Nutrition Information and Grocery Shopping

In order to address the challenge of allowing users to explore heterogeneous data sources to build knowledge, we apply our pivoting approach to the domain of food and health. In this exploratory example we pivot across a nutritional information data source, and an online grocery store source. We have identified two websites to be used as pivotable data sources: *Ocado*, an online grocery delivery website; and *NutritionData.com*, a web site that lists the nutritional content of a wide variety of foods.

Ocado is a modern online grocery shopping web site that allows users to purchase fresh grocery food over the internet. The web site provides comprehensive categorisation of products using a detailed deep hierarchy, as illustrated in Figure 4.18, which depicts the options available to further filter down the 26 products in the “Apples” category. Additionally, the screenshot shows the top ranked product in this category, “Ocado Everyday Apples,” which provides the name of the product and the price, not shown in the screenshot is that products available in multiple sizes have the weight specified.



Shop / Food / Fruit, Veg & Salad / Fruit /

Apples 26 products

On Offer (3)

Fridge (13)
Cupboard (13)
Dried (1)
Fresh (24)

Free From (12)
Organic (4)
Vegetarian (15)

Braeburn (4)
Bramley (1)
Fuji (1)
Gala (4)
Golden Delicious (2)
Granny Smith (3)
Jazz (1)
Pink Lady (4)
Red Delicious (1)
Sundowner (1)

Multipack (1)
Sliced (1)

Waitrose (20)
Ocado (6)

← Previous 1-20 Next →

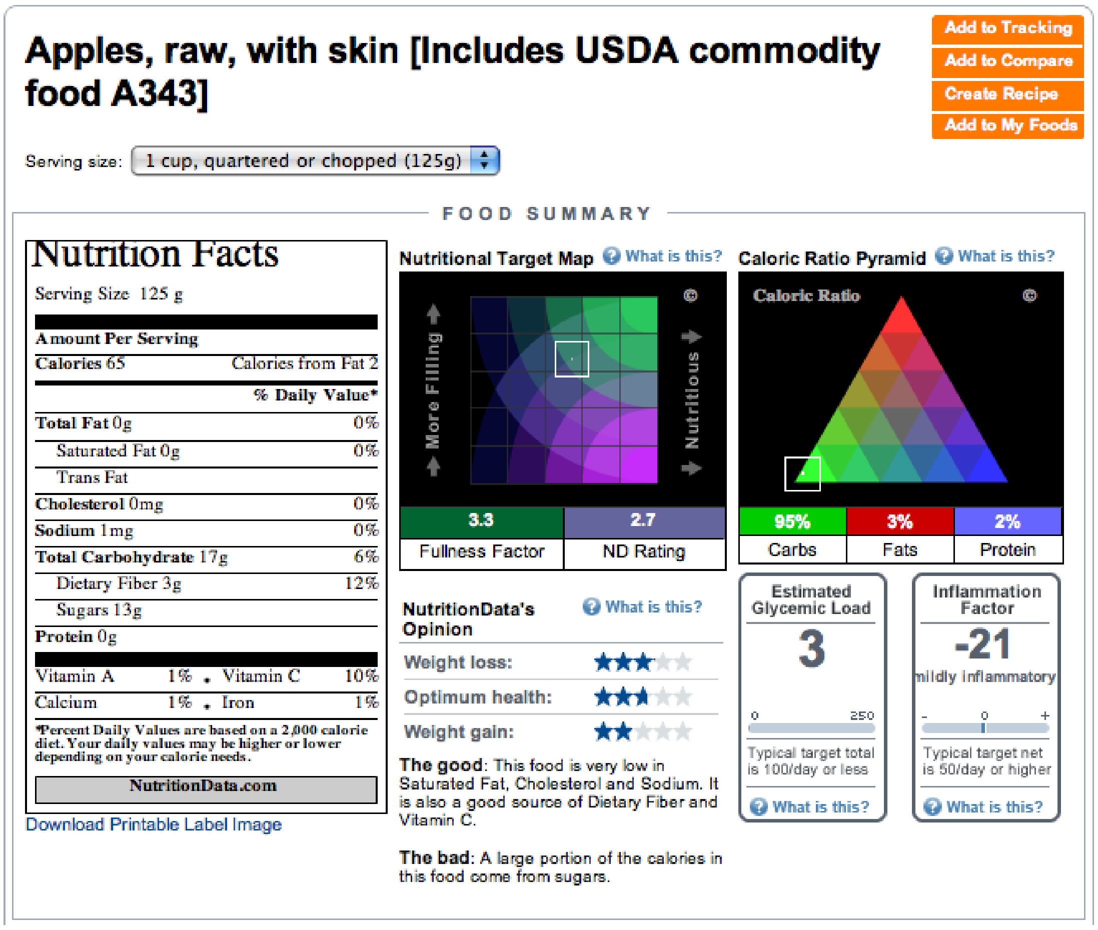
Ocado Everyday Apples
99p
Life 7 days guaranteed (average 7.5)*

FIGURE 4.18: Screenshot of the *Ocado* website, where the user has browsed to "Food", to "Fruit, Veg and Salad", and finally to "Apples." The facets of Apples to further drill-down are shown, as well as an example product.

NutritionData.com is a website that specialises in producing detailed nutritional information about different fruits, vegetables, ingredients and meals. The site mimics the standardised table design found on many food products, as depicted in Figure 4.19.

We consider these web sites to be a good platform as an indication of useful data sources as they both have:

1. A large number of visitors.
2. A large number of different products.
3. A wide variety of foods.
4. A wide variety of different metadata on products.
5. A natural overlap in products.
6. Metadata that can be web-scraped.



Apples, raw, with skin [Includes USDA commodity food A343]

Serving size: 1 cup, quartered or chopped (125g)

FOOD SUMMARY

Nutrition Facts

Serving Size 125 g

Amount Per Serving	
Calories 65	Calories from Fat 2
	% Daily Value*
Total Fat 0g	0%
Saturated Fat 0g	0%
Trans Fat	
Cholesterol 0mg	0%
Sodium 1mg	0%
Total Carbohydrate 17g	6%
Dietary Fiber 3g	12%
Sugars 13g	
Protein 0g	
Vitamin A 1%	Vitamin C 10%
Calcium 1%	Iron 1%

*Percent Daily Values are based on a 2,000 calorie diet. Your daily values may be higher or lower depending on your calorie needs.

NutritionData.com

Download Printable Label Image

Nutritional Target Map

3.3	2.7
Fullness Factor	ND Rating

Caloric Ratio Pyramid

95%	3%	2%
Carbs	Fats	Protein

NutritionData's Opinion

Weight loss:	★★★☆☆
Optimum health:	★★★☆☆
Weight gain:	★★☆☆☆

The good: This food is very low in Saturated Fat, Cholesterol and Sodium. It is also a good source of Dietary Fiber and Vitamin C.

The bad: A large portion of the calories in this food come from sugars.

Estimated Glycemic Load

3

0 250

Typical target total is 100/day or less

Inflammation Factor

-21

mildly inflammatory

- 0 +

Typical target net is 50/day or higher

FIGURE 4.19: Screenshot of the *NutritionData.com* website, where the user has searched for "apples" and browsed to "Apples, raw, with skin". The summary of the nutritional information for the food is shown, for a volume of 1 cup of apples.

7. Complementary uses; adding prices to nutritional information, or nutritional information to shopping adds value.

The sites have a natural semantic pivot on food products, for example, *Ocado* sells Chicken Breasts, and *NutritionData.com* has a page providing the nutritional information of Chicken Breasts. In the next section we describe how we aligned nutritional information with products, how the data was collected from the web and which metadata fields from the sources we use.

We find further motivation for the task of finding healthy food, from online shopping comparison sites that attempt to offer users the option of swapping out food from their basket for others that lower attributes of the food, such as the calories or fat content of the food. For example, the MySupermarket site[12] allows users to "swap out" cheaper versions of food to save money, and goes further by also offering to swap out low calorie, low fat, low saturated fat, low salt and low sugar alternatives, across four supermarkets.

[12]MySupermarket: An online grocery shopping comparison site. `http://www.mysupermarket.co.uk/landingpages/healthy-shopping.aspx`

As this is one of the motivating examples that our pivoting approach enables, the fact that this has been implemented over real data indicates that there is a perceived consumer demand for it. The use of our genericised approach allows users to go further with filtering food, and thus further addresses needs that a user may have, such as foods with low carbohydrates or suitable for those on a wheat-free diet. Likewise, other datasets can be included in our approach, so that users can also compare food that is *fair trade/ethical* or *organic*. Furthermore, not all users are looking to reduce calories, a body builder, for example, may wish to find higher calorie options for their meals, which they could do with a faceted pivoting approach.

b

The following facets have been defined in our faceted interfaces, from the data that was available from the sources. In the *NutritionData.com* set, many more facets are available (such as specific types of fat content and the levels of vitamin content for many types of vitamin), however these were not all included, as they are not necessary for our intent and the purpose of the prototype. Specifically, we extracted the following fields from each source:

Ocado :

Product type This facet describes the type of product, which maps roughly to the aisle in the supermarket, for example “Cooking Oils” and “Pizzas.”

Product name The name of the product, for example “Chicago Town Take Away Stuffed Crust Pepperoni Pizza” and “Odysea Meze Roasted Red Pepper.”

Price (in pence) How much the item costs, in British pence.

Package size The size of the product, for example “160g.”

NutritionData.com :

NutritionData.com Product This facet shows the name of the product from *Nutritiondata.com.*

Calories The amount of calories (in kcal), per 100g (or 100ml for liquids) serving of this product.

Total Fat The amount of fat (in grammes) per 100g of this product.

Total Fat: Saturated The amount of saturated fat (in grammes) per 100g of this product.

Calcium The amount of calcium (in mg) per 100g of this product.

Protein The amount of protein (in mg) per 100g of this product.

Riboflavin The amount of riboflavin (in mg) per 100g of this product.

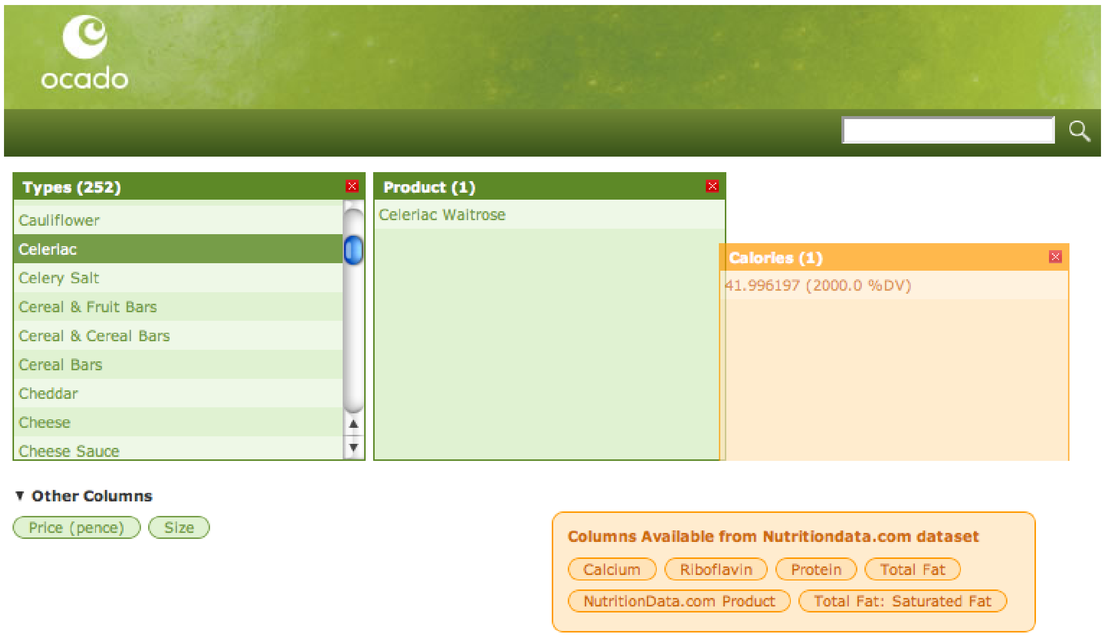


FIGURE 4.20: Screenshot of our pivoting mSpace prototype, showing a user *pivot-pull*ing a column into their current browsing session.

An illustration of the two sources pivoted in a single interface is shown in Figure 4.20.

We describe our observations of the implementation of this pivoting interface in the next section.

4.8.10 Pivoting Prototype: Observations and Issues

Semantic pivoting presents a novel technique of providing users with on-demand cross-domain matching through their faceted browser. We have noted the following observations that are unique to semantic pivoting:

1. Multiple matches means that multiple attributes are offered when an item is selection, e.g. when the user selects "Chocolate Orange Bar" in the *Ocado* browser, and alignments have been made with "Chocolate Orange Milkshake" and "Terry's Chocolate Orange," attributes from both of these matches will be shown to the user. For example, when the user pivot-pulls the "Calories" facet into the slice, they will be offered the values for the milkshake and the chocolate orange, where the milkshake isn't that close, but the chocolate orange is similar enough (by weight, at least). In some situations there will be right and wrong values, in some there will be close-enough values, in some, it might be appropriate to find an average of the values. Either way, provided that the user themselves is able to make an informed human decision about what information is being provided to them (in this case the provenance of the calorie values), then the pivot has been useful. In

any case, the fact that the system has prevented them from having to find out the matches themselves manually, it has been a success.

4.9 Knowledge Building On-The-Go — mSpace Mobile

In this section we detail our solution to building knowledge on the move, which we called "mSpace Mobile." We investigated knowledge building in the mobile context to further explore faceted browsing under more restrictive conditions. Specifically, we investigated knowledge-building, where users wish to ask queries related to the their location, such as nearby restaurants, things to do, and public transport times.

The mSpace Mobile interface allows concurrent visualization of these kinds of queries, while also presenting options for selection, and information about any object of current interest, while enabling a quick gesture to add a chosen item to a list for future reference. It is also possible to see ratings by friends or trusted peers of any items in view. With GPS ghost, there are also reminders available about places one may have visited previously. The persistent, concurrent availability of relevant related information reduces cognitive load and memory overload by enabling recognition rather than recall:

4.9.1 The Gestalt of mSpace Mobile

The heart of mSpace Mobile interaction is the mSpace interaction model (m. c. schraefel et al., 2005) for exploratory search with faceted browsing (Hearst et al., 2002): this model imagines information as a high dimensional space. High dimensional spaces are hard to visualize, so mSpace imagines a projection onto the space, flattening it to create a series of slices on the space. These slices create, effectively, a temporary hierarchy of the subset of dimensions in the domain. The dimensional slice is represented as a multi-column viewer, where selections of the elements (or facets) in the left column act as filters on elements in the right column. As mSpace Mobile is location sensitive, a map is used for the preview cue mechanism. The mSpace Mobile interface version of the mSpace interaction model is designed to support users of small screen devices carry out three main activities: (1) the querying of large information spaces through direct manipulation; (2) the rapid assessment of those spaces for information of value; and (3) the ready capture of information for later assessment. Cathy Marshall calls such rapid assessment "information triage" (Marshall and Shipman III, 1997). Abigail Sellen points out that this kind of information gathering and assessment activity takes up a higher percentage than any other Web-based knowledge worker activity, and yet is the least supported in browsers (Sellen et al., 2002). Problems on the desktop with assessing and then keeping track of information assessed over pages of clicks is only amplified on limited screen devices. To support information triage on small devices, mSpace Mobile combines the following interaction techniques, getting significant benefit over any one of these techniques used alone: These are: ZedPanes, Dimension selection, Triage Space, Annotator/Recommender, and GPS Ghost. We continue by describing the interface layout.

4.9.2 Interface Layout

The central focal point of the mSpace Mobile work is the user interface, which is laid out using a focus+context (Rao and Card, 1994) style of layout. In this type of layout, a number of panes show different information on the small screen, and expand when tapped, so that the user can alter the focus of their work. We illustrate the interface in Figure 4.21, and detail the individual panes in Figure 4.22.

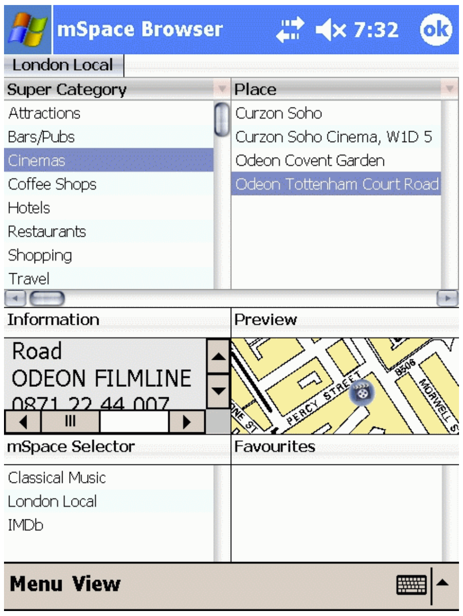


FIGURE 4.21: Screenshot of our mSpace Mobile prototype, showing the multi-pane interface, with an mSpace faceted browser at the top, and separate information and map panes below.

4.9.3 ZedPanes

ZedPanes is a modified fisheye lens type zoomable, multipane spatial interface specifically designed to support persistent focus+context interaction in mobile devices. The ZedPanes interface builds on DateLens (Bederson et al., 2004). It uses the same kind of fisheye zoom effect to enable peripheral, legible views of information while enabling focus areas to be enlarged. While ZedPanes also provides 3 levels of zoom like DateLens it has a unique reusable component architecture for the zooming interface that does not restrict the developer to basic table layouts. In the mSpace Mobile application while the screen shown has been divided into 5 specific zoomable areas, these areas can be easily changed around

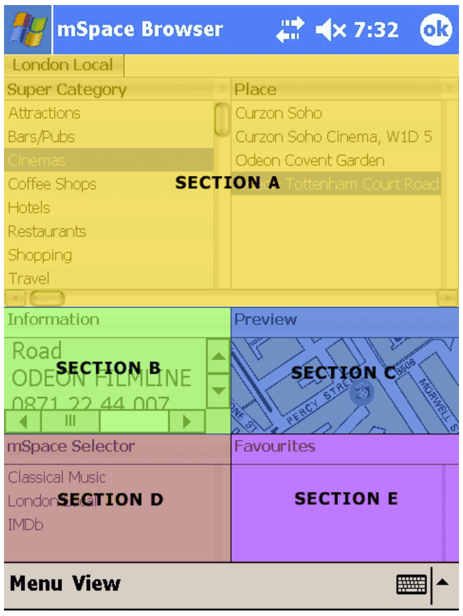


FIGURE 4.22: Screenshot of our mSpace Mobile prototype, showing the overlays on areas of the interface: Section A — the columnar entity selector; Section B — the information box; Section C — a context graphic, in this case a map; Section D — an mSpace selector and Section E — an Interest list, in this case labelled Favourites.

and reorganized on demand; additional panes can also be added. The ZedPanes approach in the case of mSpace Mobile supports the persistent availability of (a) an information selection space; (b) an information view for more detailed information on selected items; (c) a map view for locating the selections; (d) a triage area to track what has already been identified for further investigation; and (e) a selector area for shifting to related domains of information relevant to the current selection (see Figure 4.23). In mSpace Mobile, we combine Zedpanes' zooming with mSpace's rapid reorganization/selection of areas of interest (described below) to support persistent context for rapid information triage in mobile contexts.

4.9.4 Dimension Selection, PaneSlider and Multiple Selection

Dimension Selection enables alternative dimensions to be added or dimensions currently in the slice to be removed. For instance, a person may be looking at Events, Artists, Places in the top view (called a slice). They might decide remove Artists and replace it with Producers /production companies to support different interests (who is putting on the latest run of Guys and Dolls? Are they doing any other shows in town?). Similar to Dimension selection, PaneSlider enables dimensions to be moved around within the slice. This manipulation means that people can control how they wish to frame a hierarchy of interest in the information view. One may wish to view a space by Actors then Films rather than Films then Actors, as a simple example. Dimension selector and PaneSlider enable easy reconfiguration of slices with available dimensions. mSpace Mobile also enables people to select more than one entity at a time, thus enabling compound queries:

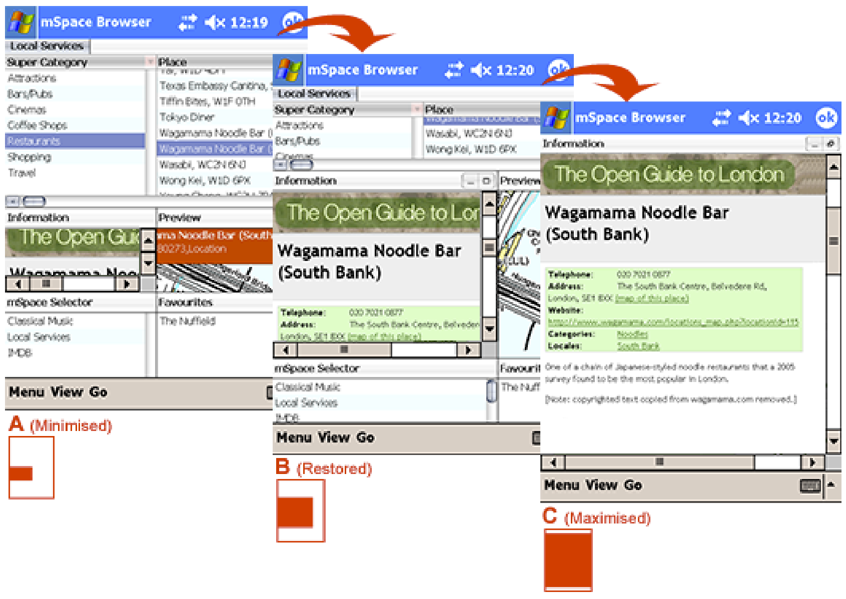


FIGURE 4.23: mSpace Mobile ZedPanes: **A** shows the default equal zoom level; **B** shows the info box expanded to the take up more of the screen but with other areas of the screen still visible; **C** shows the info box pane expanded to full screen. Any of the panes can be expanded or contracted in this way by a single click.

"show me the restaurants and theaters that are nearby". These will both be displayed in the infobox and on the map view context. This simple but powerful query technique is not yet possible in Google maps.

4.9.5 Triage Space

As people find elements of interest — such as a restaurant, a film, a CD or a gallery — they can tap and hold to add them into the Interest pane. The list of entities in the Interest pane enables real time monitoring of interest space growth, as well as an easy ability to keep elements of interest in one place, persistently in view. Selecting an entity in the list populates the info view pane with the information about that selection.

4.9.6 Annotator/Recommender and GPS Ghost

How does one choose from a possible array of alternatives? mSpace Mobile incorporates a lightweight means to view several levels of ratings from buddies, trusted community and third party sources. Icons show averages of these sources. If one is interested in interrogating a review further, they can click on the icon to explore the reviews for that entity. Likewise, a tap gesture brings up the rating/annotating box to enable rapid assignment of a rating. A person can then make a further annotation in text if they wish, either then, or later at their soonest convenience. GPS ghost is a reminder service, especially for those on the go who may not recall if they have visited a particular locale before. The GPS ghost symbol shows up on items that have been visited previously. The user can set the length of time at a site that determines a "visit." Ghosts can be collected in their own view to put together an ad-hoc diary for later annotation of locations visited.

4.9.7 mSpace Mobile Overview

mSpace Mobile showed that a focus+context interface enabled contextual faceted browsing to be performed on-the-go, under the constraints of a mobile handheld system. We were able to apply our scalable high-performance faceted browsing engine to the mobile platform to enable users to find local knowledge

4.10 Contributions and Conclusion

In this chapter we have described methods and prototype implementations for enabling browsing across heterogeneous data sources. In order to address the challenges of enabling users to build knowledge across heterogeneous data, we have presented the following contributions:

- **mSpace Faceted Browser Framework** The mSpace faceted browser framework enables users to browse faceted data through a web browser interface. Developers can assert heterogeneous data from multiple sources into an RDF triplestore, and provide a faceted interface over the data. The framework is scalable and responsive and forms the basis our later work (the contributions of which are described below) where extensions to the framework were made to support simpler setup, user selection of source data, user creation of mSpace interfaces, alignment of data from multiple mSpaces, and pivoting across domains using mSpace.
- **Facet Ontology** In order to inform the faceted interface which data it should populate the faceted interface with, a description of an abstracted view of that data is required. We contribute Facet Ontology, which is used to describe the structure of data that should be use to populate faceted browsers. Facet Ontology is an OWL ontology that allows RDF classes and predicates to be specified as relationships between data that should be put into facets in an interface. Facet Ontology allows transformations of data to be specified, so that when data is harvested, it is filtered through different algorithms before it is rendered and seen by the user. For example, a date facet might be filtered to just a "year," in order to create a Year facet. The use of Facet Ontology means that a model of a domain can be shared, because it is written in RDF, and therefore has a URI, which is

resolvable using the methodologies of Linked Data. By sharing descriptions of models, they can be extended and used as a basis for further interfaces of the same, similar or overlapping domains that use the same data source(s).

- **mSpace Maker** To translate a Facet Ontology description into a working faceted browsing interface, a data extraction tool must be used. Our contribution of mSpace Maker performs this task with respect to the creation of faceted interfaces using the mSpace framework. mSpace Maker takes the URI of a Facet Ontology definition, and uses it to extract knowledge from data sources, and populates an mSpace database with that knowledge. A basic mSpace, ready to be customised and styled, is then deployed over that mSpace database. The submitter of the mSpace to the maker is then e-mailed with the URL, and is able to browse the data specified in the Facet Ontology definition. The mSpace Maker is extensible so that additional data transformations can be added, that conform to third-party ontologies.

- **Data Picking Interface** One of the barriers to entry in creating a faceted interface over a data source is being able to explore the source data, and accurately describe the semantic relationships between data. In order to enable all users, and not just those that have experience in the Semantic Web, to be able to pick data from a semantic graph, we contribute the Data Picker. Our tool accepts an entity URI or a SPARQL endpoint URL as its input, and displays data within that graph in minimally technical way. Specifically, we render the data as a scrollable column, as it would look if it were included in a faceted browser. This approach therefore shows users what they are adding to an interface by accepting a particular class of data into their browser. The Data Picker takes a lightweight graph walking approach to picking data, where classes of data available are shown to the user. The user picks a class of interest, and all datatype properties (i.e. all of those that lead to literals) are then rendered as column, and added to the faceted interface. The user can then select any they with to exclude. The default-include nature of the interface is intentional, as it allows users to browse through large scale data sources adding a number of facets without having to select them all individually. When the graph walker encounters any object properties (i.e. predicates that lead to other objects) they are then shown in the graph walker, and all literals of the linked object class are then added to the faceted browser also. The exploration is then a recursive repetition of the above actions, until the user decides they have all of the data they require. The data picker then creates a Facet Ontology definition of the data, and submits this to the mSpace Maker, so that the user can have access to a faceted interface straight away, without the need to submit their definition to the mSpace Maker or handle the definition in any way, if they do not with to.

- **Facet Portal** One of the benefits of our Facet Ontology approach to describing data spaces is that the definition can be shared and built on by others. In order to simplify this process, we contribute the Facet Portal service. This service offers a registry of Facet Ontology definitions, which it indexes so that users can search for interface definitions that match their domain of interest. The portal registry also allows faceted interfaces and data pickers to query the indexes in order to find overlapping domain models, to aid with extending existing models, and pivoting from one model to another.

- **Semi-automated Facet Semantic Alignment Service** Our alignment service contributes a lightweight mechanism for semi-supervised co-reference of entities from multiple sources. Entities from data sources described in Facet Ontology definitions are aligned using the Alignment API (Euzenat, 2004) (originally designed for aligning OWL classes, but re-purposed for our faceted alignment). Domain experts can then alter the threshold of alignment, in order to reduce false-positive and false-negative matches from the aligner. The alignments are used to create the pivot between two data sources.

- **Pivoting Faceted Browsing** We contribute pivot support for faceted browsing, so that users of a faceted interface can search for additional domains, and drag in facets from those domains in order to filter their current browsing interface using those facets. Support for pivoting in mSpace provides a lightweight interaction technique that utilises the information on the facets held in a facet portal, so that users can take advantage of facet definitions that exist, and browse across domains, through pivots.

In the next chapter we present the validation methodologies that we used to validate the solutions presented in this chapter.

Chapter 5

Validation

In this chapter we detail the validation we applied to our solutions that were presented in the previous chapter. We validated our work using the following methodologies:

1. **Formal User Studies** In order to test the efficiency and efficacy of our interaction research we performed formal user studies to compare our methods against existing work.
2. **Tasks Now Made Possible (that were previously impossible)** We have validated some of our work by demonstrating that it enables tasks to be performed that were not previously possible. By applying our work to areas where users have previously not been able to do things, and then allowing them to do it, we validate our solution. When validating in this way we specify the
3. **Engineering Tractability** Similar to the above, we also validate some of our work on the tractability of the engineering approach. Specifically that we have engineered a solution that is tractable, reusable and solves a problem that was not tractable to engineer prior to our research.

We begin by detailed the formal user studies that we performed.

5.1 Formal User Studies

In this section we detail the formal user studies we produced on the following systems:

1. **mSpace Mobile** In this user study we tested the effectiveness of the mSpace Mobile interface compared to the state of the art under both stationary and in-motion conditions.

2. **Backwards Highlighting** This user study tested the effectiveness of backwards highlighting against bucket highlighting and against not having any highlighting. We tested discovery of facts, retention of facts, affect on behaviour and understandability.

5.1.1 mSpace Mobile

In this section we detail the results of our user study that compared mSpace Mobile to the state of the art approach in both stationary and mobile conditions. Our hypothesis that mSpace Mobile would perform more effectively in each condition was borne out. Figure 5.1 shows the completion times for each participant in each condition and UI. The lines across the graph show the average completion time for each UI in each condition. Paired t-tests were used to evaluate the specific differences between the two dependent variables, the mobile and stationary conditions of each interface. As we were not interested in any effect between interfaces, paired t-tests (rather than ANOVA) were sufficient for a comparison of two means. mSpace Mobile performed significantly faster (30%, $p<0.0005$, $t=6.5566$) than Google Local in the stationary condition; this increased to almost 40% faster, also significant ($p<0.0001$, $t=12.2425$), in the in-motion condition. The difference between motion conditions in mSpace Mobile is not significant (6%, $P=0.6279$, $t=0.5040$), whereas the performance drop in Google Local between conditions is greater (10%), but not quite significant ($P=0.0528$, $t=2.2714$). The degree to which mSpace Mobile performed better in particular in the mobile condition, however, is a conservative value. Participants were halted after 12 minutes. This stopping value was used for subsequent statistical evaluation. Whereas all these participants in both stationary and mobile conditions with mSpace finished the tasks, 4 of 9 (44%) in the stationary and 7 of 9 (78%) in the mobile condition with Google Local did not complete the sequence by the 12 minute mark, near double the time needed to complete the sequence in either condition for mSpace Mobile. Even for a small sample size, the consistent degree of difference in terms of performance across participants and between the two interfaces, particularly in the in-motion condition, suggests that evaluating mobile devices both in-motion and with sequential tasks are an effective metric for assessing mobile UI performance.

The time required to load external pages requested from within Google Local had an effect on performance in both conditions. Each click in Google Local is a call out to the Web, which can increase interaction time, depending on network performance. In mSpace Mobile information associated with a selection, such as the next column entities, their map locations and information views is transported in smaller chunks and cached: calls to the network are reduced, overall interface response is faster. While performance time for mSpace Mobile across conditions was largely similar, Google Local's performance dropped considerably from the stationary to the mobile condition. While network performance remained equivalent between conditions, it became apparent from observing participants

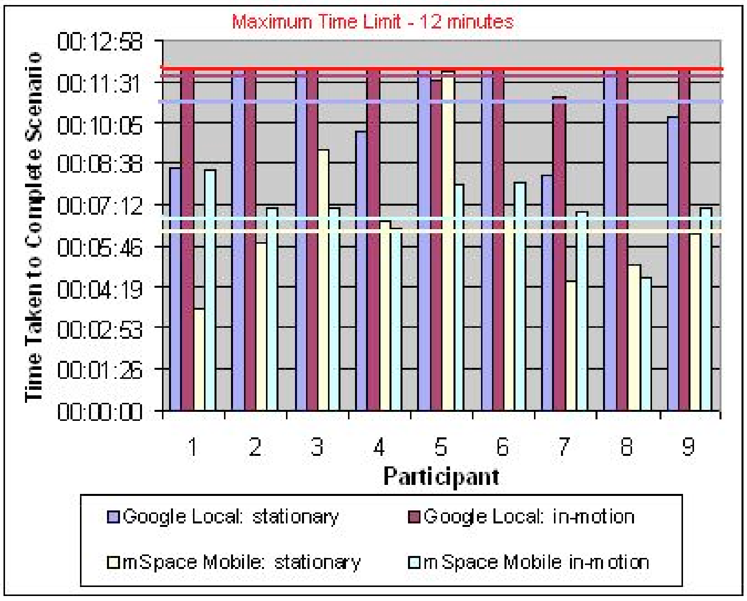


Figure 5.1: Graph showing the performance times of each participant in the four conditions.

that scrolling and text entry in Google Local took longer to carry out while mobile than while stationary; this supports previous work (Mizobuchi et al., 2005), (MacKay et al., 2005).

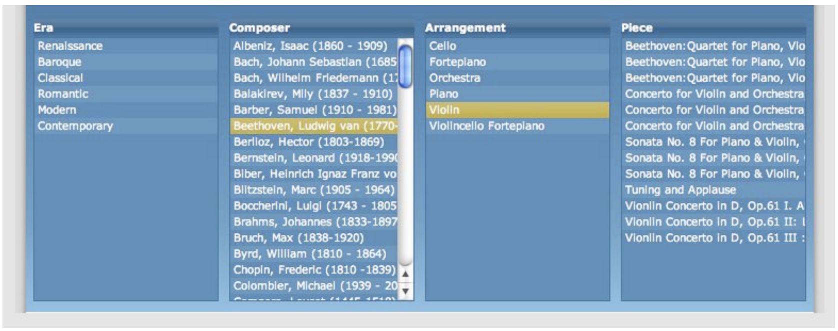
Participants commented on this difference themselves during interviews. We also noticed that participants frequently slowed their pace when entering text in Google Local, whereas there was less pace slowing observed with mSpace Mobile. One participant noted that target acquisition was challenging for selecting individual items within the columns of mSpace Mobile, though this did not seem to have a noticeable effect on their performance between the conditions. In cases where participants knew the location of something, they preferred Google Local for its text entry. For this reason, some participants said they would appreciate a text search box in mSpace Mobile as a complement to the UI (this has since been added). Overall participants said they preferred the direct manipulation of mSpace Mobile.

From the above, several conditions emerge which contribute to effective performance when carrying out planning activities with mobile devices, particularly when on the move: persistent views of information, quick data transfer, reduced requirement for text entry, and reduced requirement for activities like scrolling that require both acquiring and holding a target — this later point reinforces the findings on mobile target acquisition (Crossan et al., 2005). mSpace Mobile's emphasis on single tap selection and expandable panes reduces the need either for scrolling or text entry, contributing to improvement in performance by reducing the number of taps to the interface. While it is possible that walking amplified the scrolling and text entry problems to such an extent as to account for the considerable performance difference between conditions in Google Local, it may be that additional cognitive load factors come into play as a result of the cumulative delays caused to task completion by any one of these factors, reducing performance further. In contrast, the performance of mSpace Mobile remained fairly constant across conditions.

This finding suggests that mSpace Mobile's non-page paradigm for presenting Web data, with resulting reduced calls to the network, its largely persistent views of information in a domain, and its focus+context interface may reduce cognitive load and improve performance by improving recognition rather than recall in the interface, particularly when on the move. Further study will be needed to tease out these factors' effects.

5.1.2 Backwards Highlighting

In this section we describe the results of our evaluation of backwards highlighting, against bucket highlighting, and no highlighting. We compare discovery and retention of facts under the different conditions. The evaluation of the Backward Highlighting technique was designed to evaluate whether conveying this additional information has benefits for users by encouraging incidental learning. Subsequently, the conditions and measures are specifically designed to examine the effect of the additional information conveyed by backward highlighting on incidental learning. First, to attribute any incidental learning directly to information that is highlighted or filtered in the facets, a cut down simple set of directional column facets were created, as in our control condition shown in Figure 5.2 in the related work section above, which shows a Classical Music dataset.



Era	Composer	Arrangement	Piece
Renaissance	Albeniz, Isaac (1860 - 1909)	Cello	Beethoven:Quartet for Piano, Vio
Baroque	Bach, Johann Sebastian (1685	Forteplano	Beethoven:Quartet for Piano, Vio
Classical	Bach, Wilhelm Friedemann (1	Orchestra	Beethoven:Quartet for Piano, Vio
Romantic	Balakirev, Mily (1837 - 1910)	Piano	Concerto for Violin and Orchestra
Modern	Barber, Samuel (1910 - 1981)	Violin	Concerto for Violin and Orchestra
Contemporary	Beethoven, Ludwig van (1770-	Violincello Forteplano	Concerto for Violin and Orchestra
	Berlioz, Hector (1803-1869)		Sonata No. 8 For Piano & Violin,
	Bernstein, Leonard (1918-199		Sonata No. 8 For Piano & Violin,
	Biber, Heinrich Ignaz Franz vo		Sonata No. 8 For Piano & Violin,
	Blitzstein, Marc (1905 - 1964)		Tuning and Applause
	Boccherini, Luigi (1743 - 1805		Vionlin Concerto in D, Op.61 I. A
	Brahms, Johannes (1833-1897		Vionlin Concerto in D, Op.61 II:
	Bruch, Max (1838-1920)		Vionlin Concerto in D, Op.61 III :
	Byrd, William (1810 - 1864)		
	Chopin, Frederic (1810 -1839)		
	Colombier, Michael (1939 - 20		

FIGURE 5.2: A simple directional column-faceted browser with the Arrangement column filtered by the Beethoven and the Piece column filtered by Beethoven and by Violin Arrangements. This interface is used as the control condition in the user study described below.

The user never leaves this view, and no further information is given except for what is shown, highlighted, and filtered in the columns. The chosen columns may not be rearranged, removed, or added to. Our first experimental condition, and our initial conceptual design for the BH technique, highlights related items in facets to the left of selections Figure 5.3. Highlights are differentiated to selections by colour, where selections are yellow and highlights are green. In addition, a count of highlighted items is displayed at the top to indicate whether there are any highlights that are currently out of view and can be seen by scrolling. After a selection is made, any parent facets automatically scroll until a highlight is visible; this may involve no scrolling if one is already in view. In this paper, we refer to this first experimental condition as the Backward Highlighting condition.

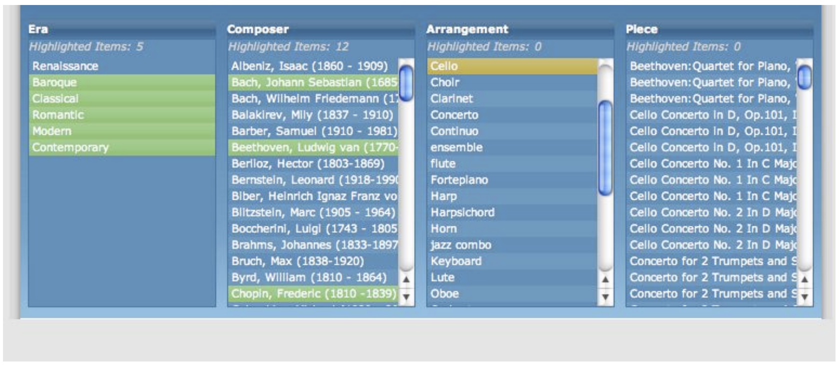

Era
Highlighted Items: 5
Renaissance
Baroque
Classical
Romantic
Modern
Contemporary
Composer
Highlighted Items: 12
Albeniz, Isaac (1860 - 1909)
Bach, Johann Sebastian (1685
Bach, Wilhelm Friedemann (17
Balakirev, Mily (1837 - 1910)
Barber, Samuel (1910 - 1981)
Beethoven, Ludwig van (1770-
Berlioz, Hector (1803-1869)
Bernstein, Leonard (1918-1990
Biber, Heinrich Ignaz Franz vo
Blitzstein, Marc (1905 - 1964)
Boccherini, Luigi (1743 - 1805
Brahms, Johannes (1833-1897
Bruch, Max (1838-1920)
Byrd, William (1810 - 1864)
Chopin, Frederic (1810 -1839)
Arrangement
Highlighted Items: 0
Cello
Choir
Clarinet
Concerto
Continuo
ensemble
flute
Forteplano
Harp
Harpsichord
Horn
jazz combo
Keyboard
Lute
Oboe
Piece
Highlighted Items: 0
Beethoven:Quartet for Piano,
Beethoven:Quartet for Piano,
Beethoven:Quartet for Piano,
Cello Concerto in D, Op.101,
Cello Concerto in D, Op.101,
Cello Concerto in D, Op.101,
Cello Concerto No. 1 In C Maj
Cello Concerto No. 1 In C Maj
Cello Concerto No. 1 In C Maj
Cello Concerto No. 2 In D Maj
Cello Concerto No. 2 In D Maj
Cello Concerto No. 2 In D Maj
Concerto for 2 Trumpets and S
Concerto for 2 Trumpets and S
Concerto for 2 Trumpets and S

FIGURE 5.3: The first experimental condition called 'Backward Highlighting'. In the columns, the Cello Arrangement is selected and the Piece column shows all the Pieces with a Cello Arrangement. The Era and Composer columns show the items associated with Cello music with highlights.

The second experimental condition, shown in Figure 5.4 represents an alternative design, which is identical to the first experimental condition, but also copies all highlighted items into a separate 'bucket' at the top of each column. This separation is much like the dual menu designs presented by (Sears and Shneiderman, 1994). Then regardless of the position of the scrolled list, the highlighted items persist in view and can be scrolled independently of the column if needed. While there is a necessary trade-off for screen real-estate, this condition is designed to identify any significant advantages for using this space to group highlighted items. The items in the 'bucket' can be selected by the user, as in the column itself, and causes the same behaviour. In this paper, we refer to this second experimental condition as the Bucket Highlighting condition.

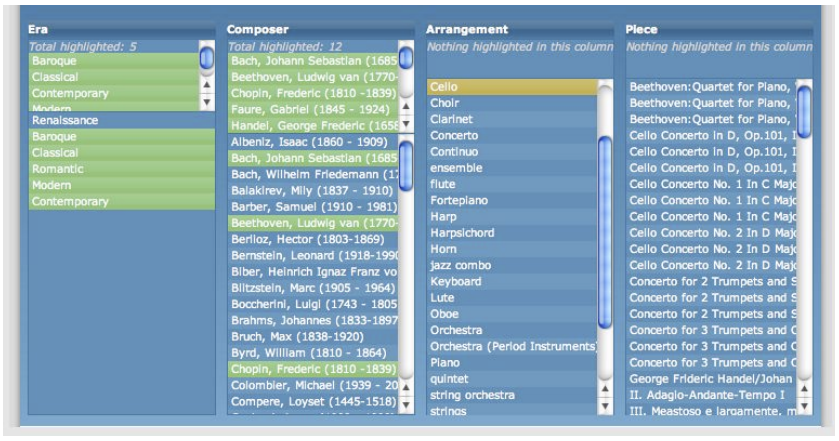

Era
Total highlighted: 5
Baroque
Classical
Contemporary
Modern
Renaissance
Baroque
Classical
Romantic
Modern
Contemporary
Composer
Total highlighted: 12
Bach, Johann Sebastian (1685
Beethoven, Ludwig van (1770-
Chopin, Frederic (1810 -1839)
Faure, Gabriel (1845 - 1924)
Handel, George Frederic (1658
Albeniz, Isaac (1860 - 1909)
Bach, Johann Sebastian (1685
Bach, Wilhelm Friedemann (17
Balakirev, Mily (1837 - 1910)
Barber, Samuel (1910 - 1981)
Beethoven, Ludwig van (1770-
Berlioz, Hector (1803-1869)
Bernstein, Leonard (1918-1990
Biber, Heinrich Ignaz Franz vo
Blitzstein, Marc (1905 - 1964)
Boccherini, Luigi (1743 - 1805
Brahms, Johannes (1833-1897
Bruch, Max (1838-1920)
Byrd, William (1810 - 1864)
Chopin, Frederic (1810 -1839)
Colombier, Michael (1939 - 20
Compere, Loyset (1445-1518)
Arrangement
Nothing highlighted in this column
Cello
Choir
Clarinet
Concerto
Continuo
ensemble
flute
Forteplano
Harp
Harpsichord
Horn
jazz combo
Keyboard
Lute
Oboe
Orchestra
Orchestra (Period Instruments)
Piano
quintet
string orchestra
strings
Piece
Nothing highlighted in this column
Beethoven:Quartet for Piano,
Beethoven:Quartet for Piano,
Beethoven:Quartet for Piano,
Cello Concerto in D, Op.101,
Cello Concerto in D, Op.101,
Cello Concerto in D, Op.101,
Cello Concerto No. 1 In C Maj
Cello Concerto No. 1 In C Maj
Cello Concerto No. 1 In C Maj
Cello Concerto No. 2 In D Maj
Cello Concerto No. 2 In D Maj
Cello Concerto No. 2 In D Maj
Concerto for 2 Trumpets and S
Concerto for 2 Trumpets and S
Concerto for 2 Trumpets and S
Concerto for 3 Trumpets and C
Concerto for 3 Trumpets and C
Concerto for 3 Trumpets and C
George Frideric Handel/Johan
II. Adagio-Andante-Tempo I
III. Meastoso e largamente, m

FIGURE 5.4: The second experimental condition called 'Bucket Highlighting'. The same information is shown as in Figure 5.3, except that the highlights in the Era and Composer columns are copied and collected into a separate scrolling list above the column. The user may now scroll through only the highlighted items.

The study was carried out with three hypotheses regarding the effect of BH on incidental learning:

H1: We hypothesised that having BH would permit incidental learning on parent facets. Parent facets are those presented to the left of facets that contain the subject of learning and thus are parents in the left-to-right hierarchy. While we expected it to be true that using the BH will allow users to learn about parent facets where they could not before, it is important to check that this is true and that simply highlighting has the same effect as filtering. In particular we expected to see a significant effect between the two experimental conditions and the control condition.

H2: We hypothesised that grouping any backward highlighted items, as in the Bucket Highlighting condition, would aid incidental learning and make the technique more effective. Given that highlights are as effective for incidental learning as filtering, we suspected that collecting them together will emphasise this ability to learn incidentally from the columns. In particular we expected to see a significant increase in incidental learning between the two experimental conditions.

H3: We hypothesised that the result of increased incidental learning on parent facets would lead to an altered interaction behaviour in subsequent interactions with the columns. That is to say, rather than scrolling through a long list to find an item in the middle of a path through the columns, participants will begin to alter their patterns of interaction to start at the beginning of a path and find information using filtering. We believe that, being a more efficient method of accessing the data, users will be able to improve as their knowledge of the information space increases.

In the remainder of this paper we will refer to this Altered Pattern of Behaviour as APB. In particular, we expected to see a significant rise in the number of re-finding tasks carried out with APB between the control condition and, potentially, both control conditions.

The study was a within-groups repeated measures design, where participants were exposed to each of the three conditions in a counter-balanced order. Each participant began by filling out an agreement and demographic survey. There were 18 participants and so each condition was trialed 18 times, and each order of exposure was trialed 3 times. The participants ranged between 18 and 65 years old, 12 were male and 6 were female, and they had varying levels of education and subject backgrounds. With each condition, the user carried out 2 task-sets, which are described below. After using all 3 conditions, participants were asked to carry out some memory tasks, also described below. Finally, each participant was debriefed with a semistructured interview regarding the tasks and their experiences with the conditions. Participants were given a music voucher in appreciation of giving their valuable time.

To reduce learning effect between the conditions, three datasets were used in the study:

a classical music dataset, a historic news film dataset, and a BBC TV programming catalogue dataset; the association of dataset and condition was also rotated. Each dataset had four facets. The facets of the classical music dataset were Era, Composer, Arrangement, and Piece. The facets of the news film dataset were: Theme, Subject, Topic and Story Title. The facets of the TV programming dataset were: Subject, Series, Contributor, and Title. Having multiple datasets also provides opportunity to evaluate any subtle differences if a significant pattern is found between them in analysis.

Two task-sets were carried out with each condition, which involved one learning task and three re-finding tasks. One task-set focused on an item in the second column and the other focused on an item in the third column. This separation of task made sure that there was always one filtered child facet and one highlighted parent facet, while supporting a comparison of the two in later analysis. Finally, every participant carried out a memory task for each task-set and each condition, leading to a total of six task-sets and six corresponding memory tasks.

Learning Tasks. In line with the hypothesis of the evaluation, the tasks were exploratory and involved the user learning about items in the middle two facet columns. As we are measuring learning effect, the tasks have to be very carefully planned so that they do not have an effect on what is learnt. (Klauer, 1984) presented a meta-analysis of a decade of incidental learning studies, concluding that incidental learning is impeded by giving behavioural objectives, learning directions or questions before an instructional text. Subsequently, questions we very clear and simple, with no added context. An example task, given simply a facet and an item within it, was: “Please learn about the Arrangement: Orchestra”.

Re-finding Tasks. Based on what was discovered during their learning tasks, participants were asked to carry out three re-finding tasks. Each time the condition was reset to its starting position, so that the behaviour within each refinding task had the same baseline and did not vary depending on the previous task.

Memory Tasks. After both task-sets had been completed on each condition, the participants were asked to recall as much information as possible about what they had learned during the initial learning tasks. These were carried out in the same order as the task-sets within each condition, so that there was a gap between learning and recall in all cases.

From there we intended to measure the amount of incidental learning in three ways. First, the user was asked to write facts about each object of the learning tasks as they interacted with the browser, in a similar manner to the study carried out by Todd on the learning skills of children Todd (2006). Within this written account, the number of statements made about items in the parent facets to the left of the learning objects were counted. Second, we measured incidental learning through their interactions, by asking them to refind information from their written account. Through these re-finding tasks, we measured the amount of interaction with facets to the left of the facets containing the objects of the tasks. We hypothesised that greater and more accurate interaction with these facets on the left indicate that the user has learned that they can use this metadata to filter down towards the requested information. This expected behaviour is that of users who know what they are looking for and how to find it, which is in line with the expectations of many existing search techniques. Finally, the users were asked to complete a memory task, where they try to write down as much as they remember about each of their previous learning tasks. Counting the balance of statements from this second written account will measure the amount of retained information. We hypothesised that backward highlighted items have a similar retainment in memory as filtered items.

We consider each of our hypotheses and the results that support or disprove them. We finish with a discussion of additional interesting results found during the study.

H1: Highlights Support Incidental Learning As hypothesized, there was a statistically proven increase in the number of parent facts learned and remembered by participants in the experimental conditions over the control condition and there is no significant effect on child facts. Figure 6 shows the number of parent facts and children facts written down during learning and the number of parent and child facts written down during the memory tasks.

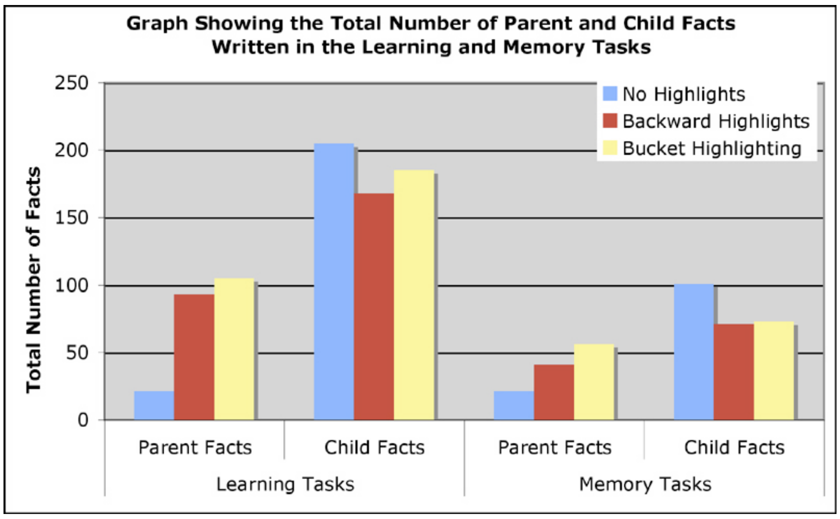


FIGURE 5.5: Shows the total number of facts written at different stages of the study, by condition, where Parent Facts are those written about facets to the left of a selection, and Child Facts are those written about filtered facets to the right. As expected, no significant difference was found in the Child Facts between any of the conditions. The differences seen in the Learning Parent facts is significant (ANOVA, F=14.97, $p<0.0001$), as are the Memory Parent Facts (ANOVA, F=5.292, $p<0.01$).

In the learning tasks, the trend in written parent facts is highly significant (ANOVA, F=14.97, $p<0.0001$). The significance, however, is between the control condition and the two experimental conditions, while the increase in parent facts written in the Bucket Highlighting condition, over Backward Highlighting, only has a significance of $p<0.5$ (t-test, t=0.6972).

In the comparison of all three conditions during memory tasks, the difference is still highly significant (ANOVA, F=5.292, $p<0.01$), but the amount remembered in the Bucket Highlighting condition over Backward Highlighting is more significant (t-test, t=1.5357, p=0.1336) than in the learning tasks.

While the significant effect of having highlights was expected and supports our first hypothesis, these statistics alone do not tell us about any significant difference between Backward Highlighting and Bucket Highlighting. These two specific conditions are investigated further below.

H2: There is a benefit to grouping highlights Our second hypothesis was that grouping highlights, as in our Bucket Highlighting condition, had a significant learning advantage to the user over simply highlighting items as they appear in the columns. Although there was no significant difference in the number of facts written down in the learning and memory tasks, post-study questionnaires revealed that Bucket Highlighting was the preferred condition for 16 out of 18 participants. Further to this, the significant trend (ANOVA, F=5.715, $p<0.005$) shown in Figure 5.6 shows that users found the Bucket Highlighting condition slightly easier than Backward, and Backward Highlighting condition easier than the control. This trend of decreasing difficulty matches the increasing trend in the number of Parent facts written in the Learning and Memory tasks in Figure 5.5. In Figure 5.6 we can also see that there was no significant difference in previous knowledge participants had within each condition.

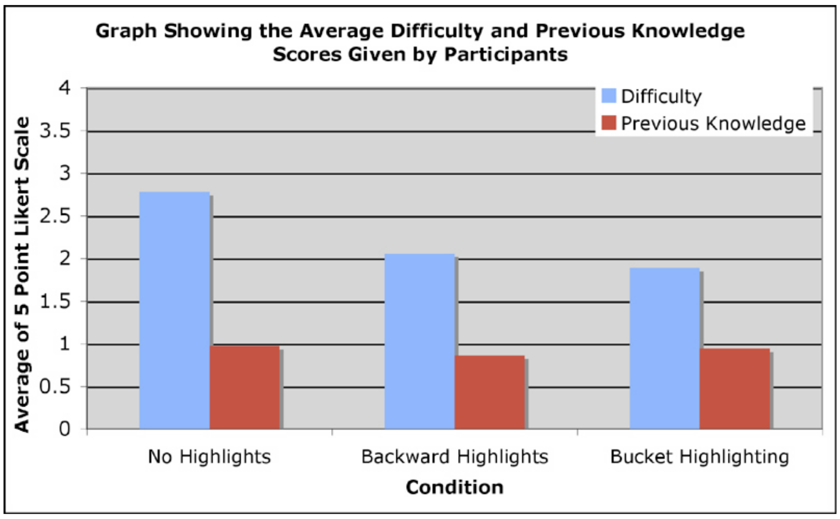


FIGURE 5.6: Average score from a Likert scale for both difficulty and previous knowledge, by condition. No significant difference was found in the previous knowledge scores, but an ANOVA showed that Bucket Highlighting was perceived as easier to use than Backwards Highlighting.

During further discussion in the structured interviews users consistently explained that they preferred Bucket Highlighting for when the list was longer, because then they didn't have to scroll to find information. To investigate this further, an analysis was carried out by condition and dataset. Figure 5.7 shows the number of parent facts written down during memory tasks for both Backward and Bucket Highlighting conditions. While there is little difference seen for the Classical and News film datasets, a significant difference (t-test, t=2.7269, p<0.05) is seen between the two conditions on the BBC TV scheduling dataset. Analysing the three datasets, we see that both the Classical Music and News film datasets have 6 and 15 items in the first column, respectively. In comparison the BBC TV scheduling dataset has over 200 items in the first column. Subsequently, we can deduce that Bucket Highlighting shows specific advantage as the size of the parent columns increase.

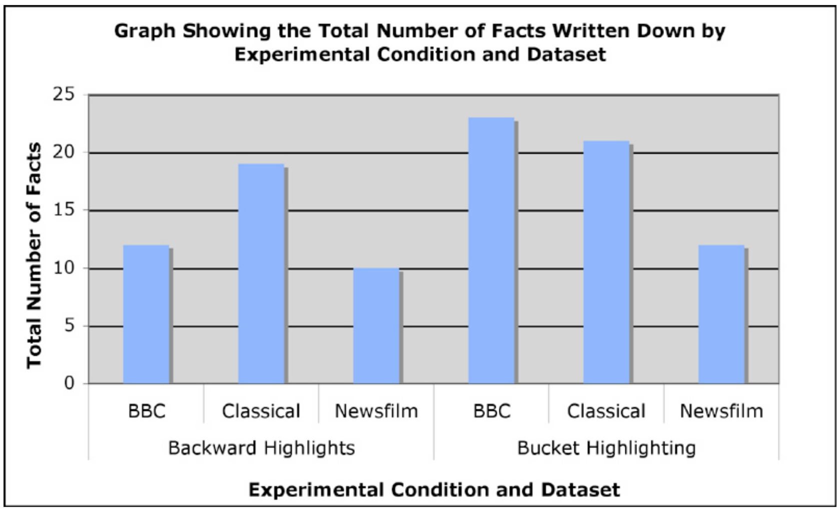


FIGURE 5.7: The total number of facts written in the memory task phase of the study, by condition and then dataset. The only significant difference is between the Backward and Bucket conditions with the BBC TV Scheduling dataset (t-test, t=2.7269, p<0.05). Analysis of the dataset shows that it has significantly longer left-hand columns than the other datasets.

H3: Highlights will improve future interactions Our third hypothesis, suggesting that highlights would allow people to improve their interaction patterns in refinding tasks, which we called APB above, was not supported by the results. Figure 5.8 shows the average number of tasks with APB in each condition. With a maximum average of three, we can see that users very rarely altered their pattern of interactions, opting instead to repeat their previous patterns. Most users, who did alter their patterns of interaction, did so consistently, but no significance could be obtained from this graph or from any other analysis by dataset or by demographics as to what causes this effect. There is a slight gender imbalance, as shown by Figure 5.9, but this was only significant to p=0.35 (t-test, t=0.9221). Interestingly, this pattern is exaggerated for younger females and older males. The most significant trend, although still low, was found by education (ANOVA, F=1.627, p=0.2), as shown in Figure 5.10, where we see a large bias towards users who have a Bachelors degree, and reduces with further education. This does not relate to age, as a separate analysis shows age effect to be insignificant. Subsequently we must reject our third hypothesis and leave the investigation of APB to future work.

These results are not conclusive as to the cause of APB, but we have shown that it

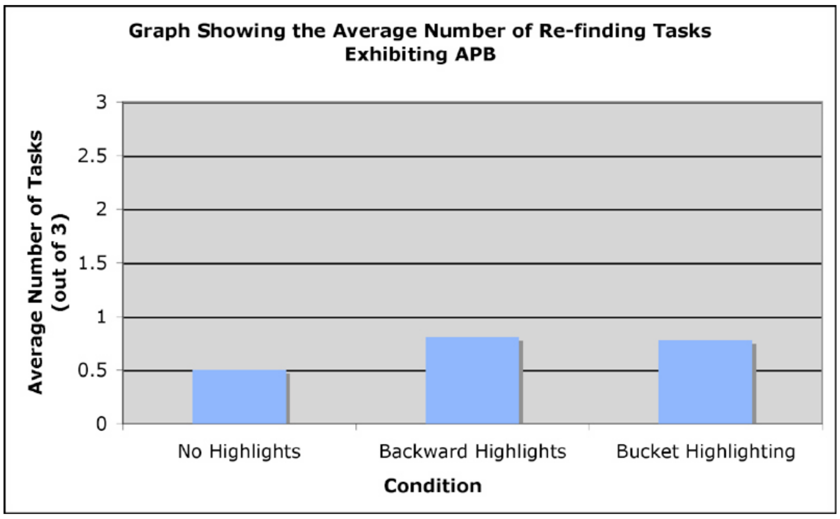


FIGURE 5.8: The average number (out of 3) of re-finding tasks with APB, by condition. No significant difference was found by condition.

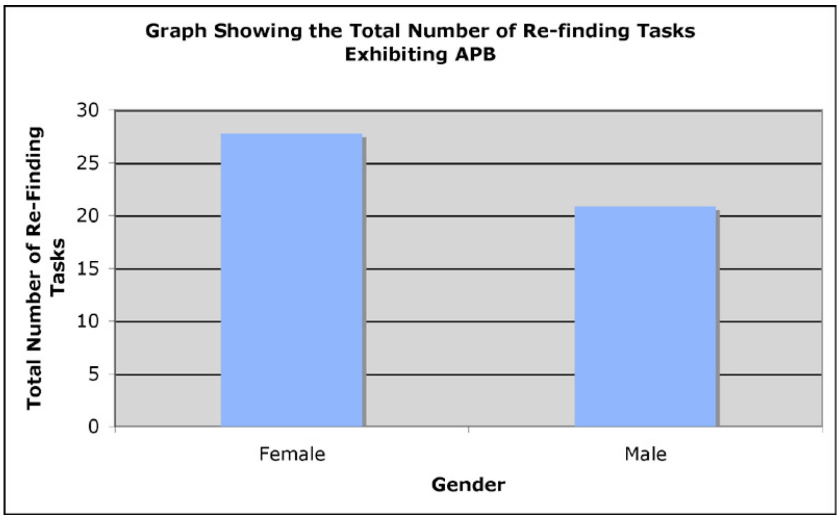


FIGURE 5.9: The total number of re-finding tasks with APB, by gender. A larger trend is seen, but is still not significant.

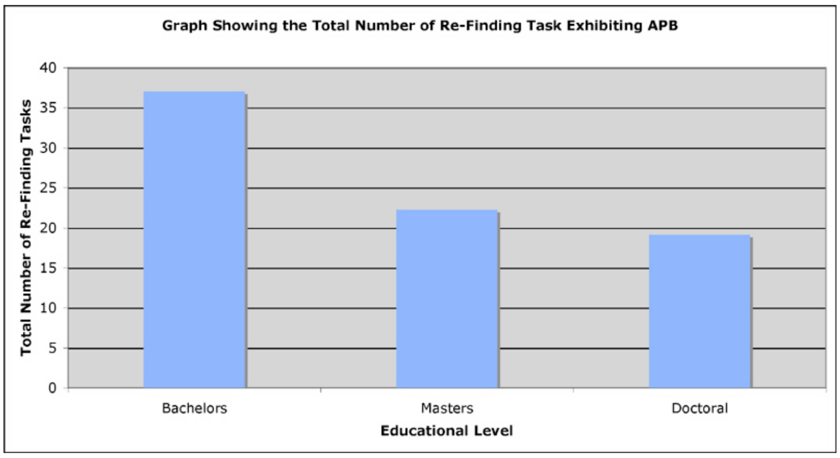


FIGURE 5.10: Total number of re-finding tasks with APB, by educational level. An even stronger trend is seen towards less educated people, but is still not significant. No significant trends were found across the collected data, and requires further investigation.

is not significantly affected by the addition of highlights that might guide some users towards optional parent filters. It would be interesting in future work to investigate these behavioural differences more, but within a wider focus than one possible highlighting technique.

5.2 Tasks Now Made Possible

In this section we validate our work by detailing the tasks that have now been made possible through application of our research output. In order to aid reader understanding of the work we begin with a motivation scenario.

5.2.1 Motivating Scenario

In order to situate our work we use the following motivating scenario of a user that wants to learn about classical music, but does not know the terminology to use to search the domain. Thus, our goal is to enable data to be as useful as possible to our user, so that they can explore it, learn, leverage necessary presentation cues, and find something that they do understand in order to orient their exploration. In order to move towards this goal, we suggest that a reasonable starting question in learning about classical music is "What classical music will I like to listen to?"

An approach to learning about classical music is to listen to a specialist radio station such as Classic FM[1], where anyone can listen to a number of pieces of popular classic music. However, this approach lacks learning and additional information about the recordings that the user is hearing – who is the Composer? what is the name of the piece? – the user cannot use this approach to further their understanding of what it is they are listening to, what about that music they like, and what domain-specific terminology to use to explore further examples of this music.

Similarly, the iTunes Music Store will expose a user to thousands of recordings, but with an emphasis on promoting recordings in order to sell them to the user – through the store the user may find music that they like, but the further reading and ability to discover *why* they like particular pieces and to find more through that learning isn't available, and thus only a shallow understanding is gleaned by the user.

There is a wealth of information that has been digitised and has been published onto the web that describes different pieces of classical music, eras, works, opera and so on, but for a user with no experience in the domain, the concept of where to start and how to get to a place where they know the questions to ask is unclear. It's not useful to that user to ask any question of the domain if they have no knowledge of the domain. For

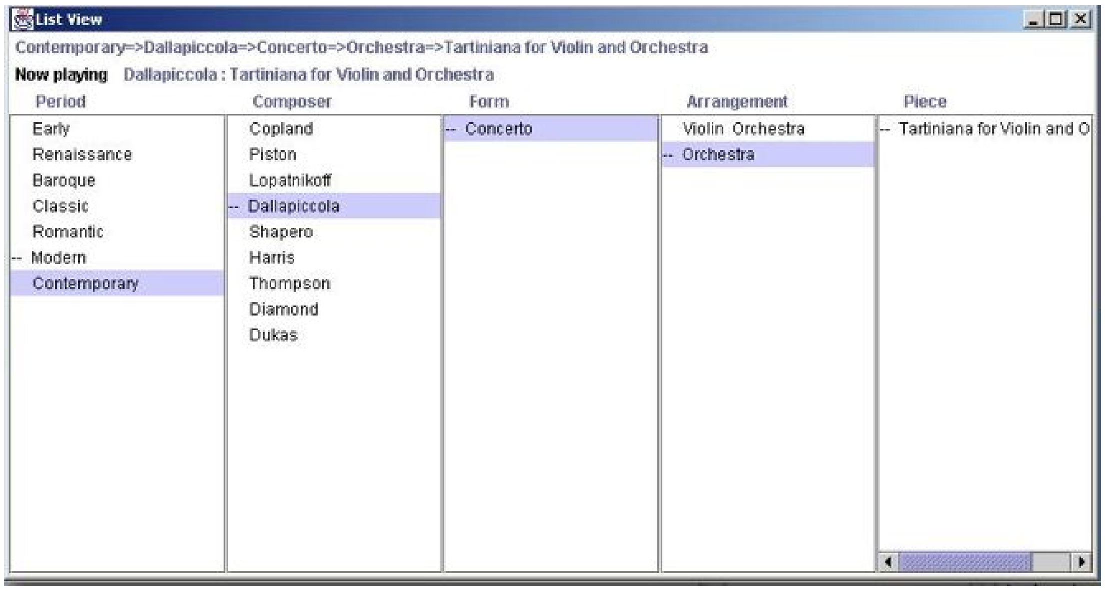
[1]Classic FM: `http://www.classicfm.co.uk/`

example, a list of music described as “Baroque” may not mean much to someone without knowledge of Classical Music.

A faceted approached to finding classical music has shown to be a good fit to this challenge (m. c. schraefel et al., 2005), since it frees users to explore a space from any point, rather than having to know the terminology to search on, or how to explore a deep hierarchy of domain-specific terminology. Likewise, it was shown that using various media, such as audio preview cues (m.c. schraefel et al., 2003), aids in faceted exploration of an unfamiliar data space.

In the case of Classical Music, a solution might be to provide a good user interface that lets users explore different facets of the space, such as via *instruments*, for example *piano*, *violin*, *viola*, or via *composer*, *era*, *style* and so on. This has implications for techniques that will enable the interface to be able to pull in the data to feed the interface, which lead to challenges in supporting data retrieval from the Semantic Web and/or using Linked Data.

Indeed, the domain of classical music has been tackled using a faceted approach, and an interaction technique known as preview cues (m.c. schraefel et al., 2003), as illustrated in Figure 5.11, where users can explore different facets of classical music, while all the time hearing previews of work that fits into those facets when they brush their mouse over the facets. The user can then direct their exploration by drilling down through the data when they hear music that they like.


List View

Contemporary=>Dallapiccola=>Concerto=>Orchestra=>Tartiniana for Violin and Orchestra

Now playing Dallapiccola : Tartiniana for Violin and Orchestra

FIGURE 5.11: Screenshot of an early faceted browser implementation from (m.c. schraefel et al., 2003), showing classical music facets that a user has begun to explore.

5.2.2 Enhancing Cross-Collection Research in Musicology using Semantic Pivoting — MusicSpace

Further to our original research we also applied mSpace to research projects. One of the largest of these projects is to enable musicologists to explore and query multiple scholarly resources. Thus, we present a musicology use case, supported by a pivot-enabled mSpace, called *musicSpace*, which contains different musicology resources that we have connected using our pivoting approach. In particular, the *musicSpace* project[2] is designed to enable musicology researchers to browse and make constrained queries across collections in a single interface; previously to *musicSpace*, making complex queries across collections was not possibly, and there was no application that enabled this.

The field of musicology research benefits from our heterogeneous browsing approach because it enables users to perform queries across musicology collections. Specifically, a number of collections have been frequently used in classical music research: Grove Music, RILM, RISM, COPAC and The British Library Sound Archive. Recently, musicologists have focused on developing these collections so that they can support research, and thus they have been catalogued and digitised. However, there has been a smaller effort to unify or align collections. Without *musicSpace*, musicologists have to manually collate information from multiple collections. Thus through applying semantic pivoting to these collections, researchers can perform complex cross-collection queries that were not possible before.

Research Challenges

Enabling cross-collection resources in the domain of musicology research presents a number of specific challenges:

1. **Data Formats:** By partnering with the musicology collections, we were able to benefit from asking them to directly send up exports of their metadata, and thus we did not have to scrape any data from their web sites. Each collection, however, utilises different technology to maintain their data, and some collections are aggregations of smaller collections. As such, the data we wanted to aggregate was provided to us in a number of different formats, and we faced the challenge of decoding and aligning a number of different formats.

2. **Schema Alignment:** Similar to the *data formats* challenge discussed above, we also faced the challenge of aligning a number of different schemas a vocabularies. Although the domain experts expended much time and effort in analysing commonalities in the data, as data sources of increasing complexity were added, earlier

[2]musicSpace: `http://mspace.fm/projects/musicspace`

decisions regarding the design of the aggregated schema, particularly decisions made with reference to simpler data sources, were often revisited to take account of unanticipated metadata types.

3. **Multi-Language Sources:** The domain of musicology often refers to works by composers that did not speak or publish in English. Preserving the original language of titles and names is highly important, as information can be lost through translation. Thus, a number of challenges were faced regarding internationalisation and translation issues. Firstly, there is the issue of equivalent term matching when performing searches. A user that searches for “Opera Buffa,” an Italian style of opera, would also benefit from matching documents that refer to “opere buffe,” which is the Italian plural of “Opera Buffa.” An English-language aware word-stemming indexer would not be able to link these two phrases, and thus the user would not be aware of all of the documents that match their search. As such we faced the challenge of how to ensure that phrases in non-English languages are properly supported.

In the sections that follow we describe the *musicSpace* project in more detail, and discuss the coverage and organisation of the musicology collections. We illustrate our prototype and then describe how our pivoting approach applies to musicology specifically. We proceed by describing how our pivoting approach aids musicology researchers.

musicSpace Project

In order to enable such advanced research, the musicSpace project has partnered with data publishers, aggregated and enriched their data, and developed a richly featured exploratory search interface to access the combined dataset. There have been several significant challenges to developing this service, and intensive collaboration between musicologists (the domain experts) and computer scientists (who developed the enabling technologies) was required.

Additionally, in many domains a single source may be considered to be definitive for certain types of information. In musicology, this is essentially the case with the “works lists” of composers’ musical compositions given in Grove Music Online (`http://www.oxfordmusiconline.com/public/book/omo_gmo`), and so for musicSpace, we have mapped all sources to the works lists from Grove for the purposes of exploration, specifically to exploit the accuracy of its metadata in respect to dates of publication, catalogue numbers, and so on. Therefore, rather than mapping all fields from Grove to a central model, it would be far quicker (in terms of development time) to create a system to “pull-in” data from other sources that are mapped directly to the Grove works lists.

Musicology Data Sources

This section presents the five music collections used in our evaluation:

1. **Grove Music Online:** A digitised version of *The Grove Dictionary of Music and Musicians*, an encyclopaedic dictionary of music and musicians, which is the largest single reference work on Western music, comprising over 50,000 articles.
2. **Répertoire International de Littérature Musicale (RILM):** A continually updated bibliography of writings on music - including books, journal articles, congress reports and dissertations. The database currently has over 500,000 records from 151 countries, and each record includes full publication details and an abstract.
3. **Répertoire International des Sources Musicales (RISM):** RISM UK holds details of the 17th- and 18th-century music manuscripts preserved in libraries and archives in the UK and Ireland. It includes manuscripts from national, public and academic libraries, county and city record offices, cathedral and chapel libraries and some private collections.
4. **COPAC:** Freely available library catalogue, giving access to the merged online catalogues of many major UK and Irish academic and National libraries, as well as increasing numbers of specialist libraries, comprising 32 million records.
5. **The British Library Sound Archive (BLSA):** The BLSA holds manysound and video recordings, with over a million discs and thousands of tapes. Its collections come from all over the world and cover the entire range of recorded sound from music, drama and literature, to oral history and wildlife sounds. Formats range from cylinders made in the late 19th century to the latest digital media.

We have chosen these specific collections because they are frequently used and hold large numbers of records. Additionally, the use cases for the sources are complementary, in that in order to find out the list of works of a particular composer, a researcher will go to Grove. To then find out where a manuscript for a particular work is stored, they will go to RISM, or to find out where a recording is available, they go to the BLSA. Thus, our approach to cross-collection browsing of heterogeneous sources cuts down the time and effort required to make connections across sources, which musicology researchers current have to do.

musicSpace Prototype: Cross–Collection Exploration

In order to support cross-collection exploration across the above described data sources we have applied the approaches described in Chapter 4. Furthermore, we have also developed and built upon techniques in order to deal with musicology-specific challenges.

Research challenges one and two are that the data from the musicology sources are in a number of different sources, and they adhere to different schemas. Thus, the key challenge facing us is how to unify these sources in a way the enables cross-collection exploration for researchers, without altering the data in such a way that loses information. In order to achieve this goal we worked with musicologists to determine what aspects of the data sources are of most importance, and to develop a single ontology that captures a unified view over the sources.

To this aim, we established a set of mappings from the schema of each individual source, to our unifying ontology. Musicologists created these mappings, in order to ensure correctness of the information, by leveraging their domain-specific knowledge. An import system that uses the mappings to import from each source is used to convert each source from its own format into RDF that adheres to the unified ontology. The RDF outputs from the import system is then aggregated in a single knowledge base. The mSpace Maker system is then used to create an explorer over the aggregated and unified data. In doing this, research challenges one and two are met.

The third research challenge describes the problem of multi-language data sources. Specifically, this is due to the pan-European nature of western classical music that is covered by the data sources, and the large amount of classical music originating in non-English speaking countries, particularly Germany and Italy. In our *musicSpace* interface there are two places that this problem is manifested:

1. **Keyword searching:** While the interface is localised only for English, it is reasonable to expect and to support users searching for domain-specific phrases in the source's native language, particularly when it is not customary to perform translations of titles or genres, as is the case in musicology. This problem is further compounded by the fact that some sources use antiquated foreign language in their phrasing. In particular, modern keyword search engines, such as Sphinx[3] and Lucene[4] utilise query expansion techniques (Jones et al., 1972) such as word-stemming, which require support for a specific language, and are particularly optimised for modern English, thus, while a word-stemming English-language search engine may know to match a document containing the term "musical" when a user searches for "music," it does not also know to match documents containing "opere buffe", the Italian-language plural of "Opera Buffa," an Italian style of opera.

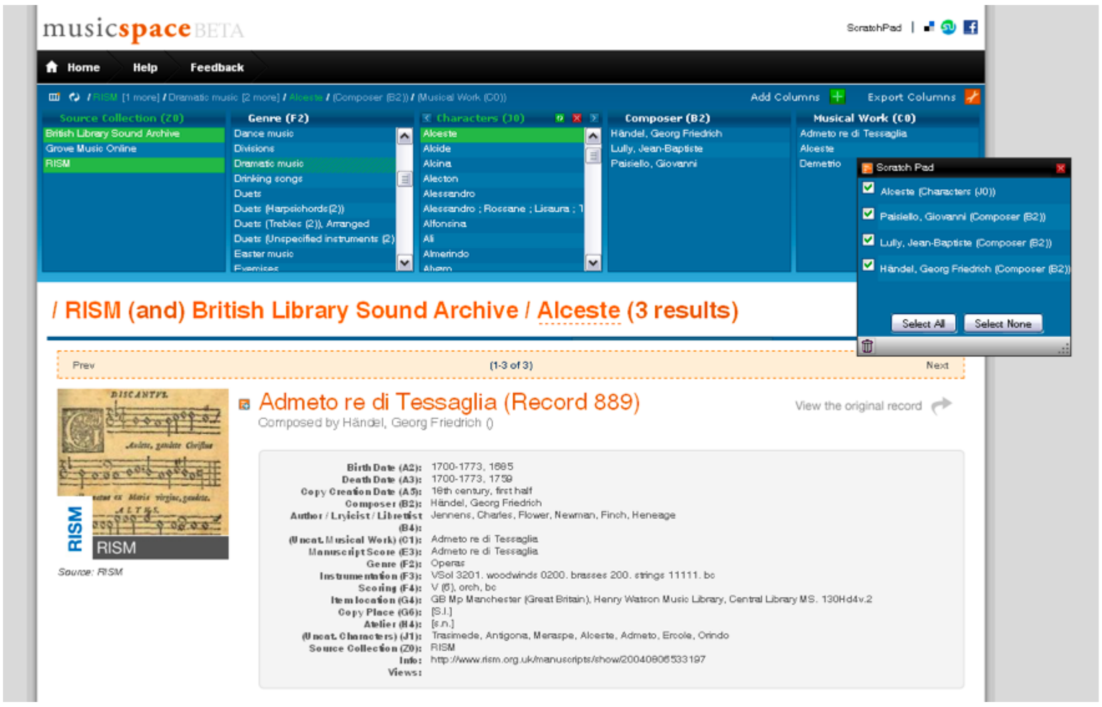
2. **Faceted browser:** Similar to the above problem, is that different data sources handle the representation of foreign characters differently. For example, we found that the german character U-umlaut "ü" is sometimes represented correctly in unicode, sometimes as an ordinary "u", and sometimes translated to "ue". These differences in representation mean that in the faceted browser multiple entries with

[3]Sphinx Search: `http://sphinxsearch.com/`

[4]Lucene: `http://lucene.apache.org`

the same word appear, and as such the composer "Heinrich Schütz" is shown also as "Heinrich Schutz", so a user would have to select both in order to see all records that relate to this composer.

A screenshot of our musicSpace prototype is given in Figure 5.12, that shows a user that selected two data sources, the British Library Sound Archive, and RISM, and has chosen the character "Alceste."


musicspace BETA
ScratchPad
Home
Help
Feedback
/ RISM [1 more] / Dramatic music [2 more] / Alceste / (Composer (B2)) / (Musical Work (C0))
Add Columns
Export Columns

Source Collection (Z0)	Genre (F2)	Characters (J0)	Composer (B2)	Musical Work (C0)
British Library Sound Archive	Dance music	Alceste	Händel, Georg Friedrich	Admeto re di Tessaglia
Grove Music Online	Divisions	Alcide	Lully, Jean-Baptiste	Alceste
RISM	Dramatic music	Alcina	Paisiello, Giovanni	Demetrio
	Drinking songs	Alecton		
	Duets	Alessandro		
	Duets (Harpsichords (2))	Alessandro ; Rossane ; Lisaura ; 1		
	Duets (Trebles (2)), Arranged	Alfonsina		
	Duets (Unspecified instruments (2))	Ali		
	Easter music	Almerindo		
	Exercises	Alvaro		

- ✓ Alceste (Characters (J0))
- ✓ Paisiello, Giovanni (Composer (B2))
- ✓ Lully, Jean-Baptiste (Composer (B2))
- ✓ Händel, Georg Friedrich (Composer (B2))

/ RISM (and) British Library Sound Archive / Alceste (3 results)

Admeto re di Tessaglia (Record 889)
Composed by Händel, Georg Friedrich ()

Birth Date (A2):	1700-1773, 1685
Death Date (A3):	1700-1773, 1759
Copy Creation Date (A5):	18th century, first half
Composer (B2):	Händel, Georg Friedrich
Author / Lryicist / Librettist (B4):	Jennens, Charles, Flower, Newman, Finch, Heneage
(Uncat. Musical Work) (C1):	Admeto re di Tessaglia
Manuscript Score (E3):	Admeto re di Tessaglia
Genre (F2):	Operas
Instrumentation (F3):	VSol 3201. woodwinds 0200. brasses 200. strings 11111. bc
Scoring (F4):	V (6), orch, bc
Item location (G4):	GB Mp Manchester (Great Britain), Henry Watson Music Library, Central Library MS. 130Hd4v.2
Copy Place (G6):	[S.l.]
Atelier (H4):	[s.n.]
(Uncat. Characters) (J1):	Trasimede, Antigona, Meraspe, Alceste, Admeto, Ercole, Orindo
Source Collection (Z0):	RISM
Info:	http://www.rism.org.uk/manuscripts/show/20040806533197
Views:	

FIGURE 5.12: Screenshot of musicSpace prototype, showing the interface in use, indicating a search for records from "RISM" and the "BLSA" with the Character "Alceste."

Pivoting Across Data Sources

In addition to the aggregated musicSpace dataset presented above, we have also produced a semantic pivoting prototype for musicSpace, where we developed the concept of "pivots" across data sources, whereby a single mapping is defined between sources so that users can browse facets from either source at once, with the browser using the semantic mapping to enable on-demand cross-referencing across sources.

This is similar to the notion of the "visual pivot" explored by (Robertson et al., 2002a), who described a system in which hierarchies are extracted from different databases, and pivot points are designated where point hierarchies intersected at common instances. For example, consider an organisation that holds a number of management databases, where a person can be contained in two different databases, a pivot point exists for each person that is in both databases. A three-dimensional visualisation is available to visually animate the transition from one hierarchy to another around the pivot point, both to show the user how the different hierarchies relate, and to enable the user to move from one related table to another to explore relationships across the data. These intersecting hierarchies are known as “Polyarchies,” and have been formally compared to the representation model of facet as used in mSpace and the structure used in ZigZag (McGuffin and m. c. schraefel, 2004), where a populated taxonomy was presented to enable comparison of the differences in representation. By using logically associated intersecting pivots in data, it is possible to query a dataset’s domain alone, and then pull in additional facets from other related sets through a semantic link.

In order for a user to browse from one faceted data set to another using pivots, a pivot-aware browser must be used. In our prototype we have used a modified version of the mSpace faceted browsing framework in order to demonstrate possible interaction methods for enabling pivoting across data sources. These interactions enable the mSpace to link automatically to other interfaces that contain pivots. These links to other interfaces are generated by a web service, and not by an individual interface. This means that they can be updated and changed dynamically without making changes to the interface. This allows links to new interfaces to be displayed in all relevant existing interfaces, and also the bi-directionality of links are preserved, therefore it will always be possible to go back to the source interface, allowing users to retrace their interaction choices. The ability to retrace their choices is important, as the recoverability of context has been shown to be beneficial to cognitive understanding during exploration (Wilson et al., 2009).

Pivoting Case Study: Musicology

In the musicSpace project, we mapped each source to a common schema, whose design was informed by the Music Ontology (Raimond et al., 2007). Specifically, from studying the available data in our combined dataset, we determined which facets could be created for our exploratory interface. During this process we encountered cases in which the common schema that we had decided upon, and which we thought would cover all eventualities of schematic representation required by our sources, needed to be altered to take account of additional metadata types found in new data sources which had not been anticipated. This necessitated the revision of the common schema, which in turn required the remapping of previously integrated sources. For example, we had specified that under a root classification of “People,” the concepts of “Author” and “Composer.” We then discovered that some data, specifically that supplied to us by Copac (`http://copac.ac.uk/`) in MODS-XML export format, did not differentiate between authors and composers, but instead referred to both as a “creator”, Therefore, in order to preserve the granularity of all data sources, we had to adjust our type hierarchy by inserting the concept of “Creator” on an additional level below “People”, and above “Author” and “Composer” (see Figure 5.13).

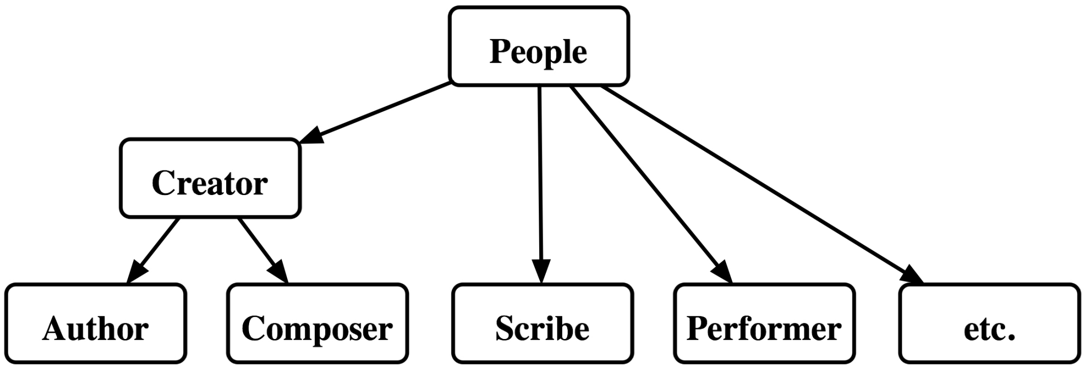
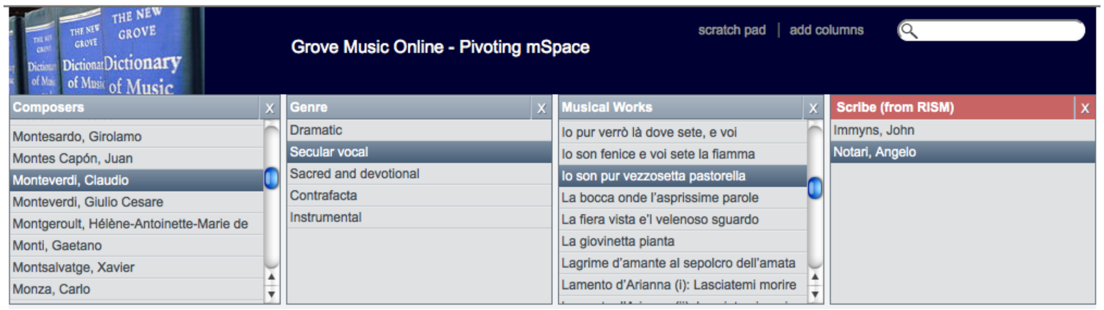


FIGURE 5.13: Illustration of the type hierarchy, with an additional level added for the "Creator" metadata.

This process would become more efficient if we could reduce the complexity of mapping to common schema, while also achieving a number of the benefits of doing so. To extend the Grove music example above, an example interaction is to search for songs in Grove, and stream them from Naxos Music Library (`http://www.naxosmusiclibrary.com/`). To do this currently, a user will search Grove by genre, retrieving a list of songs, and then cross reference that list of songs manually with recordings in Naxos, usually by querying Naxos's search interface for each song individually. By enabling a pivot on the songs in Grove with recordings in Naxos, a user can use a single pivot-enable mSpace to perform the cross-referencing for them automatically.

As a further example, consider occasions where a facet is only present in a single collection, and therefore, cannot be mapped to other sources. For instance, the Répertoire International des Sources Musicales (RISM) UK and Ireland (`http://www.rism.org.uk/`), provides metadata for each instance of an individual physical manuscript copy of a score, and thus has the facets of "Former Owner" and "Scribe" (all categories of "People"), which are not used by the other data sources that we are working with. In this case, using a semantic pivoting approach enables this cross-reference entirely, as is illustrated in Figure 5.14 by the screenshot of our prototype.



Composers	Genre	Musical Works	Scribe (from RISM)
Montesardo, Girolamo	Dramatic	Io pur verrò là dove sete, e voi	Immyns, John
Montes Capón, Juan	Secular vocal	Io son fenice e voi sete la fiamma	Notari, Angelo
Monteverdi, Claudio	Sacred and devotional	Io son pur vezzosetta pastorella	
Monteverdi, Giulio Cesare	Contrafacta	La bocca onde l'asprissime parole	
Montgeroult, Hélène-Antoinette-Marie de	Instrumental	La fiera vista e'l velenoso sguardo	
Monti, Gaetano		La giovinetta pianta	
Montsalvatge, Xavier		Lagrime d'amante al sepolcro dell'amata	
Monza, Carlo		Lamento d'Arianna (i): Lasciatemi morire	

FIGURE 5.14: Screenshot of our pivot-enabled mSpace prototype interface showing three facets from Grove Music Online, with one facet from RISM (UK and Ireland) "pulled-in" using a semantic pivot.

Feedback

Since the mSpace UI has been evaluated for exploratory search usability in a variety of contexts, our main focus in testing the musicSpace application is its impact on research: how well is it supporting the kinds of queries musicologists want it to enable? And, likewise, what new kinds of research questions, as yet unanticipated, may it enable? Towards answering these questions, we have recently completed an early pilot study. We describe our findings below. While these are early stage tests, our intention in outlining our findings here is to have knowledge of our approach and preliminary results available within the Music IR community in order to enhance engagement with the project.

A version of the musicSpace interface was released internally to a team of six musicologists for an initial period of testing and evaluation on 29 April 2009, and their feedback was very encouraging. Although this initial release did not integrate our full spread of data sources, testers nevertheless reported significant improvements with search speed and ease:

- "All the information showed up very quickly, and it was easy to find material. It was really good to have different kinds of material in the same place."
- "[musicSpace offers] a speedier way to research crossed search pathways."
- "Excellent interface — very simple to understand." Testers were also impressed with the way that music- Space's faceted interface allowed for browsing around a subject and for instantaneous paradigmatic shifts in search focus:
- "I would recommend musicSpace for its ability to manipulate queries in order to get results that you wouldn't otherwise be able to get [without starting over]."
- "I liked the ability to explore around a topic once I'd identified something of interest."
- "The ability to switch columns around and add new columns was most useful." Aside from these early hoped-for indications that musicSpace will provide a quicker and more flexible way to explore a variety of musicological data sources, testers also reported that increased search data granularity (as compared to that of our data partners' search interfaces) was a substantial benefit. For example, a number of testers were pleased by musicSpace's facility to browse by opera character:
- "[Without using musicSpace] it would not be at all easy to do a character search. You would have to use printed reference books like Pipers Enzyklopädie des Musiktheaters, but even this does not have an index of characters, so you'd have to look at the entry for each opera and draw up character lists by hand. You would also have to know what you were looking for before you started out!"

- "I used musicSpace to explore how many operas have a character named Alceste. This information simply isn't get-at-able using other search interfaces — you'd have to sort through the information on your own."

There was similar enthusiasm for musicSpace's ability to browse by scribe and the former owner of manuscripts.

Summary

Our initial approach to aggregating numerous musicological data sources for the musicSpace project relied on the manual mapping of heterogeneous data schemas by domain experts. This approach, while effective, was also labour intensive, because the central schema potentially needed to be revised as each new data source was aggregated. This raised questions as to the scalability of our approach. We have presented a technique for lowering the costs of combining heterogeneous sources that relies on the concept of Pivot-Awareness for Faceted Browsers, and we have described how this technique has been applied to mSpaces built around Grove Music Online and Naxos Music Library, and Grove Music Online and RISM (UK and Ireland).

5.2.3 Cross-Repository Browsing: Combining Multiple Heterogeneous Bibliographic Data Sources — Rich Tags

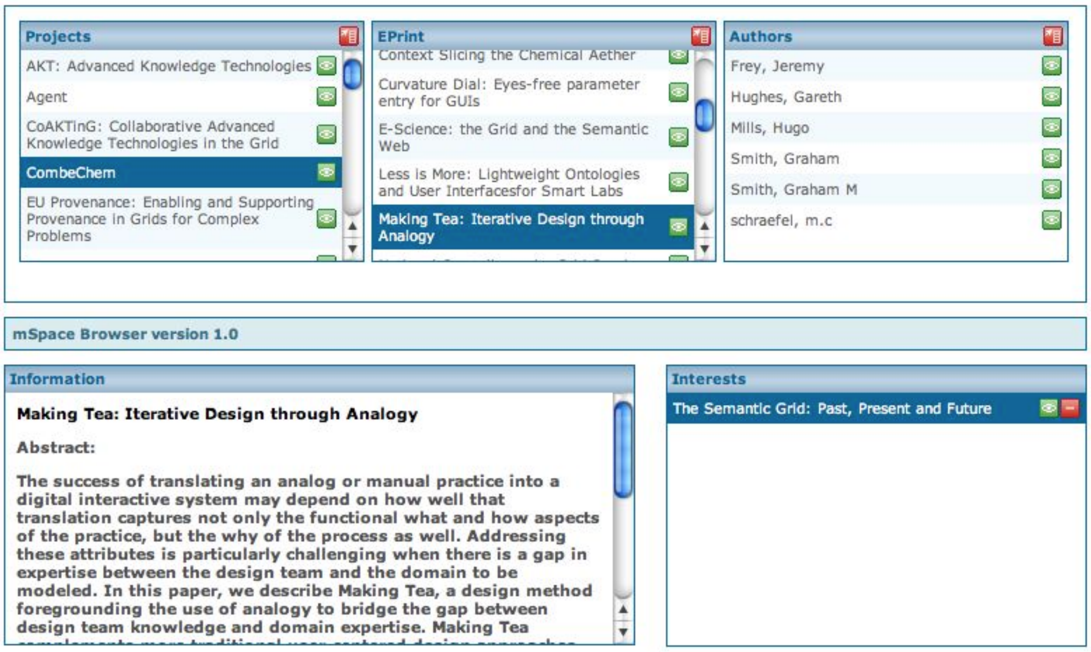
In addition to the aforementioned applications of the mSpace framework, we also applied mSpace to bibliographic data sources of open access publication metadata in a project called *Rich Tags*, in order to test our approach against data sources that dynamically update. Rich Tags looked at how we can combine metadata from multiple repositories to enable cross-repository browsing.

RichTags built on early prototypes of mSpace, which used a single source of bibliographic data, so that a digital repository could be explored. In Figure 5.15 a screenshot of mSpace running over the ECS EPrints repository is shown. This demonstrator showed that faceted browsing is beneficial for browsing metadata-rich repositories such as those used by higher education institutions. In order to expand the capabilities of this service, we went on to explore how to integrate metadata from multiple repositories.

The RichTags project utilised "lightweight inferences" in bringing together multiple sources, to enhance the browsing experience. Specifically, it did this by taking the OAI-PMH metadata feeds from multiple open access digital repositories, as well as information from DMOZ[5] and EBSCO[6]. The journals and conferences of papers were

[5]DMOZ: http://dmoz.org

[6]EBSCO: http://search.ebscohost.com


Projects
- AKT: Advanced Knowledge Technologies
- Agent
- CoAKTinG: Collaborative Advanced Knowledge Technologies in the Grid
- CombeChem
- EU Provenance: Enabling and Supporting Provenance in Grids for Complex Problems

EPrint
- Context Slicing the Chemical Aether
- Curvature Dial: Eyes-free parameter entry for GUIs
- E-Science: the Grid and the Semantic Web
- Less is More: Lightweight Ontologies and User Interfacesfor Smart Labs
- Making Tea: Iterative Design through Analogy

Authors
- Frey, Jeremy
- Hughes, Gareth
- Mills, Hugo
- Smith, Graham
- Smith, Graham M
- schraefel, m.c

mSpace Browser version 1.0

Information

Making Tea: Iterative Design through Analogy

Abstract:

The success of translating an analog or manual practice into a digital interactive system may depend on how well that translation captures not only the functional what and how aspects of the practice, but the why of the process as well. Addressing these attributes is particularly challenging when there is a gap in expertise between the design team and the domain to be modeled. In this paper, we describe Making Tea, a design method foregrounding the use of analogy to bridge the gap between design team knowledge and domain expertise. Making Tea

Interests
- The Semantic Grid: Past, Present and Future

Figure 5.15: Screenshot of an early mSpace prototype, using the ECS EPrints Digital Repository as a single source.

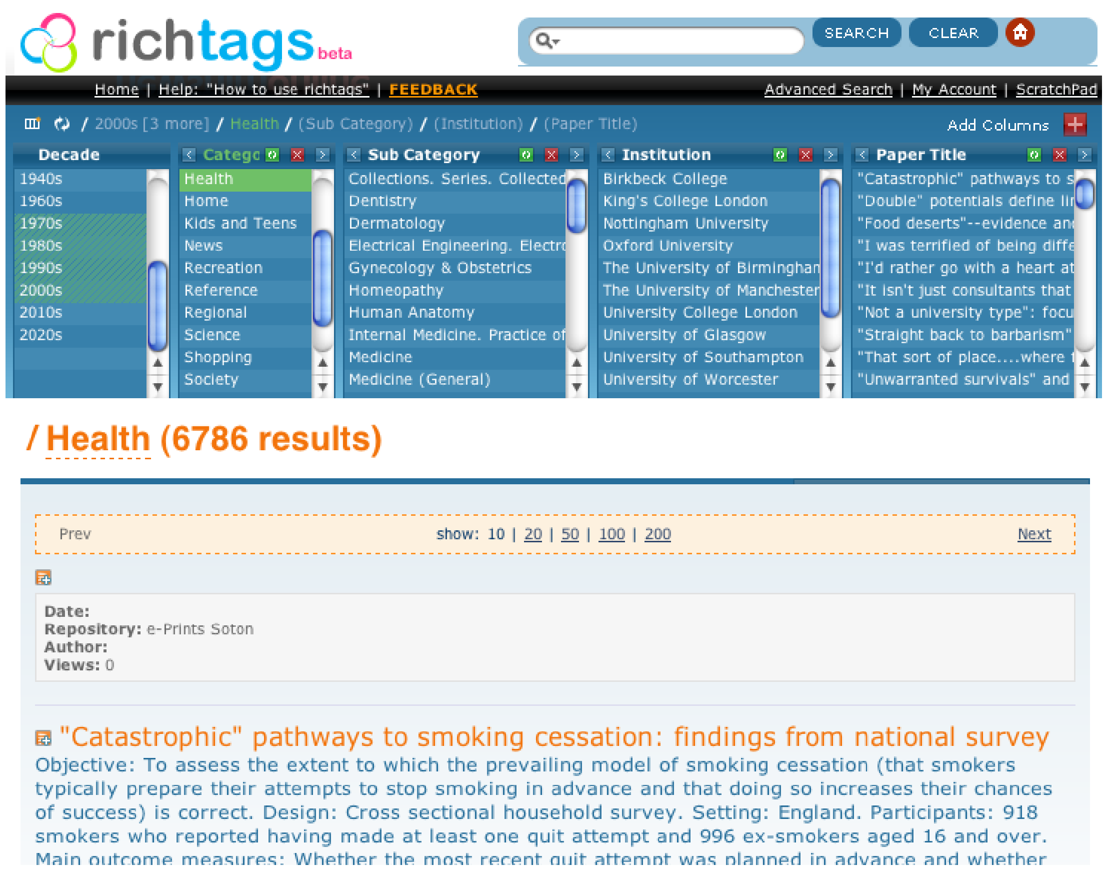
looked up in the DMOZ and EBSCO hierarchies, so that they could be applied to a two-level hierarchy of categories. This means that the mSpace explorer of Rich Tags then provides two columns “Category” and “Sub-Category” across the aggregation of all of the papers. This enables users to quickly see others papers in the same area as a paper they are looking at, that could be in another repository, even though both papers were not marked up beyond the minimal set of bibliographic information that authors add in their entries on repositories.

In the Rich Tags interface, bibliographic data from thirteen UK higher education digital repositories was harvested and used in order to demonstrate cross-repository integration using real-world data taken from a variety of sources. A screenshot of the interface in use is shown in Figure 5.16.

Future work in Rich Tags is to approach the problem of author disambiguation across repositories. We are investigating whether a lookup service from Je-S[7] (the RCUK electronic submission system, which holds information of all RCUK grant holders) would be possible and beneficial in disambiguating authors in UK-based institutions. This takes the approach of asking an external source about information.

Another approach is to look at the connections in the metadata itself, as taken by AKTiveAuthor (McRae-Spencer and Shadbolt, 2006). The AKTiveAuthor approach is to disambiguated authors by analysing self-citations across publications by the same authors. Specifically, it showed that authors tend to cite their own papers, for example

[7]Je-S: https://je-s.rcuk.ac.uk


richtags beta
SEARCH
CLEAR
Home | Help: "How to use richtags" | FEEDBACK
Advanced Search | My Account | ScratchPad
/ 2000s [3 more] / Health / (Sub Category) / (Institution) / (Paper Title)
Add Columns

Decade: 1940s, 1960s, 1970s, 1980s, 1990s, 2000s, 2010s, 2020s

Category: Health, Home, Kids and Teens, News, Recreation, Reference, Regional, Science, Shopping, Society

Sub Category: Collections. Series. Collected, Dentistry, Dermatology, Electrical Engineering. Electr, Gynecology & Obstetrics, Homeopathy, Human Anatomy, Internal Medicine. Practice of, Medicine, Medicine (General)

Institution: Birkbeck College, King's College London, Nottingham University, Oxford University, The University of Birminghan, The University of Manchester, University College London, University of Glasgow, University of Southampton, University of Worcester

Paper Title: "Catastrophic" pathways to s, "Double" potentials define lir, "Food deserts"--evidence and, "I was terrified of being diffe, "I'd rather go with a heart at, "It isn't just consultants that, "Not a university type": focu, "Straight back to barbarism", "That sort of place....where t, "Unwarranted survivals" and

/ Health (6786 results)

Prev show: 10 | 20 | 50 | 100 | 200 Next

Date:
Repository: e-Prints Soton
Author:
Views: 0

"Catastrophic" pathways to smoking cessation: findings from national survey
Objective: To assess the extent to which the prevailing model of smoking cessation (that smokers typically prepare their attempts to stop smoking in advance and that doing so increases their chances of success) is correct. Design: Cross sectional household survey. Setting: England. Participants: 918 smokers who reported having made at least one quit attempt and 996 ex-smokers aged 16 and over. Main outcome measures: Whether the most recent quit attempt was planned in advance and whether

FIGURE 5.16: Screenshot of the Rich Tags interface in use, where the user has selected Health, and 6786 papers have been found.

that when there is a paper authored by "D.A. Smith" which cites a paper by "Daniel A. Smith," those authors are highly likely to be the exact same person.

5.2.4 Facet Ontology and Pivoted Interfaces

Following our work on Facet Ontology, Data Picker and the mSpace Maker, it is now possible to achieve the following, which were not possible prior to this research:

1. Define a faceted space formally.
2. Share a definition of a faceted space.
3. Build on others' definitions.
4. Create an optimised UI from the formal specification; the optimised UI is mSpace, and has all of the features, e.g. long lists, high performance.
5. Users can make new UIs by combining multiple facet ontology definitions together, as in ocado/nutrition information example, and use automated alignment to enable mapping between the sources.

6. Users can then explore and compare data from heterogeneous sources in one interface.

5.3 Engineering Tractability

In this section we describe work that has been validated through engineering tractability. We describe how we taken wild data, and made it possible to work with it at a large scale, in order to provide users with the ability to explore very large data sets. We begin by describing the tractability when our work with mSpace moved onto wild data from the semantic web.

5.3.1 Wild Data, Larger Datasets and Geographic Data — OpenGuides Map Mashup Test Case

Our first tractability challenge was to work with "wilder" data, that is already published, in order to test our hypothesis that our model works generically for any data. Additionally, we wanted to increase the size of dataset, in order to test that our approach is scalable past small test datasets, so that it can be applied to datasets that encompass entire domains.

So far, applications of mSpace had been limited to data sets of RDF that were developed specifically for the purpose of putting into a faceted browser. This allowed complete control over how the data was structured, such that its behaviour in the mSpace environment was as intended, with a minimal amount of ontological matching or schema alignment. A test case was used to test the effectiveness of our approach in building knowledge about an unfamiliar city. Our prototype made use of third-party RDF, with OpenGuides was chosen as the source. OpenGuides is an open, community-edited wiki for building guidebooks to cities. Each wiki page can be presented as HTML for a web browser, as well as RDF containing the relevant metadata.

The OpenGuides metadata, compliant with a standard ontologies for describing, among others: name, business type, and geographical coordinates. This data could be browsed on the mSpace mobile platform, and, combined with maps, the user would be able to view places in the area.

The initial version used the established Open Guide to London. Later, the Open Guide to Southampton was founded, in order to provide a more recognisable local demonstration. These two were applied on the mSpace mobile platform, using Ordnance Survey (OS) maps integrated into the application. Lastly, a web-based mashup with the (founded by us) Open Guide to Montréal was built, for attendees of CHI2006 to explore the local area.

This used Google Maps as the flexible mapping platform, and is illustrated in Figure 5.17.

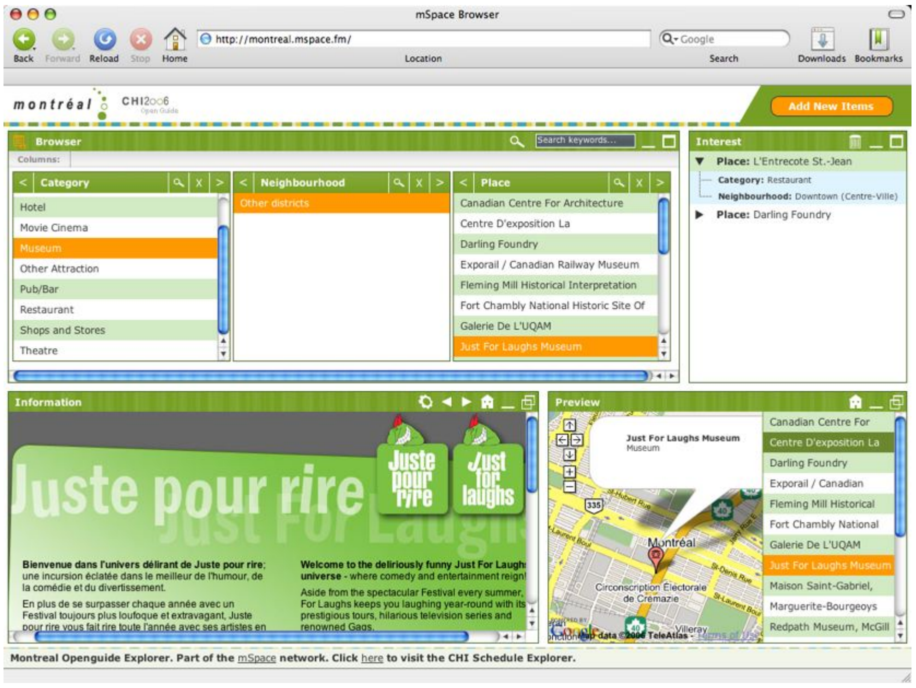

mSpace Browser
Back
Forward
Reload
Stop
Home
http://montreal.mspace.fm/
Location
Google
Search
Downloads
Bookmarks
montréal
CHI2006
Add New Items
Browser
Search keywords...
Columns:
Category
Hotel
Movie Cinema
Museum
Other Attraction
Pub/Bar
Restaurant
Shops and Stores
Theatre
Neighbourhood
Other districts
Place
Canadian Centre For Architecture
Centre D'exposition La
Darling Foundry
Exporail / Canadian Railway Museum
Fleming Mill Historical Interpretation
Fort Chambly National Historic Site Of
Galerie De L'UQAM
Just For Laughs Museum
Interest
Place: L'Entrecote St.-Jean
Category: Restaurant
Neighbourhood: Downtown (Centre-Ville)
Place: Darling Foundry
Information
Juste pour rire
Just For Laughs
Bienvenue dans l'univers délirant de Juste pour rire: une incursion éclatée dans le meilleur de l'humour, de la comédie et du divertissement.
En plus de se surpasser chaque année avec un Festival toujours plus loufoque et extravagant, Juste pour rire vous fait rire toute l'année avec ses artistes en
Welcome to the deliriously funny Just For Laughs universe - where comedy and entertainment reign!
Aside from the spectacular Festival every summer, For Laughs keeps you laughing year-round with its prestigious tours, hilarious television series and renowned Gags.
Preview
Just For Laughs Museum
Museum
Montréal
Circonscription Électorale de Crémazie
Villeray
Canadian Centre For
Centre D'exposition La
Darling Foundry
Exporail / Canadian
Fleming Mill Historical
Fort Chambly National
Galerie De L'UQAM
Just For Laughs Museum
Maison Saint-Gabriel,
Marguerite-Bourgeoys
Redpath Museum, McGill
Montreal Openguide Explorer. Part of the mSpace network. Click here to visit the CHI Schedule Explorer.

FIGURE 5.17: A screenshot of the Open Guide to Montréal faceted interface, where a map shows the geographic location of a museum, aiding the user in building knowledge of their surroundings in an unfamiliar city.

In order to fulfil our research agenda of allowing users to create interfaces over arbitrary heterogenous data, it is important that we used the data from the OpenGuides with minimal specialist processing. To this end, difficulties in working with the OpenGuides RDF soon became apparent. Previous mSpace RDF had used URIs for data as much as possible, so that two identical entities can be correctly identified. The OpenGuides RDF had a single URI for each page, however, with the properties being expressed as literals (strings, etc.). These literals are not represented on the RDF graph in the same way as a URI is (i.e. as an node on the graph), since it is not seen as a unique identifier. Such data could not be directly used in mSpace, and so it was decided to use a script to convert the RDF literals to URIs where appropriate, so that the mSpace code could use this data with minimal changes. This was a moderately heavy cost, although it was considered that this was the quickest and most extensible way to integrate the third-party RDF with the mSpace platform.

While working on the Montréal guide, further data sources were sought, and some information from various websites was scraped. It proved to be cleaner to use the OpenGuides API to post this information to the central Montréal Open Guide, before extracting the information back into RDF for mSpace. This removed the need for multiple RDF creators, and ensured the RDF returned was in a consistent format.

This test case involved a lot of heavy processing on the data, and as such there were various lessons learned. The main lesson was that it is entirely acceptable, and indeed, desirable, to utilise OWL/RDF(S) as an interchange format. By having a stable format, we could utilise semantic web tools to import the data from the guide website, without having to expunge energy and time into writing scrapers. This is a lesson that looks likely to be true for the foreseeable future. What changed, however, is that it is seen less likely to use RDF data graphs as the direct query mechanism for rich browsing in interfaces such as mSpace. Specifically we are able to support large amounts of data through traditional relational databases, compared to using triplestore technology. In this regard, there are still many challenges in the field of query optimisation, data storage, indexing and replication. These can be seen through the work such as by Virtuoso (Erling, 2006) and Oracle[8], who have demonstrated support for better query optimisation and indexing over large-scale RDF.

The results of the test cases have shown that relational database systems currently provide the best overall support for the pushing of large volumes of metadata to the interface, and that for our purposes there is no reason why the source of this metadata cannot be RDF from external sources.

In the next section we further our experimentation in browsing the Semantic Web by testing knowledge building in a mobile environment.

5.3.2 Testing Knowledge Building On-The-Go — mSpace Mobile Test Case

We hypothesised that a valuable context for Semantic Web data would be in a mobile context and with a user interface that privileged fast data access. We were keen to understand how our approach would need to be refactored, if at all, for the constraints of mobile data use. Thus, in order to test the effectiveness of the mSpace interface while on-the-move, a test case was used to compare the ability to build knowledge of a prototype *mobile* interface, against current methods of online knowledge gathering. The motivating scenario for this work imagines a user that is in a city (in the test cases, both London and Montréal were used), and has some free time between appointments, and wishes to either find somewhere to eat, with various criteria (such as desiring Japanese food), or wanting to watch a movie, and needing to find out the nearest cinema that has it, and how to get there. The test case also functioned as the first ever test of exploration of RDF data on a hand-held mobile platform (Wilson et al., 2005).

There were numerous challenges to this project, regarding both the data integration and the presentation to the limited capabilities of the mobile device. Mobile devices typically have limiting hardware and environmental factors compared to desktop and

[8]Oracle Semantic Technologies:
http://www.oracle.com/technology/tech/semantic_technologies/index.html

notebook computers, in the form of display, processing and input hardware limitations, and limited speed and availability of network connectivity. This means that virtually all components of the system have to be either specially tailored or entirely redesigned in order to function well when going mobile. We also wanted to consider another challenge, beyond supporting the platform — we wanted to consider real usability on the go, as to date, there has been very little testing of sensemaking on the go.

In order to apply this framework so that our research challenge of effective information exploration on-the-move, our existing approaches were modified to better enable retrieval and assessment of knowledge, through rapid information triage (Marshall and Shipman III, 1997). To better lay out the interface we utilised the approach of focus plus context (Rao and Card, 1994), which splits up the interface into dynamically resizable areas, that have the benefit of retaining context, while also allowing individual areas to be resized as focus as different areas are used, which is illustrated in Figure 5.18.

The challenge of exploring heterogeneous data sources is particularly difficult on mobile UIs when multiple searches are required to support a compound query.

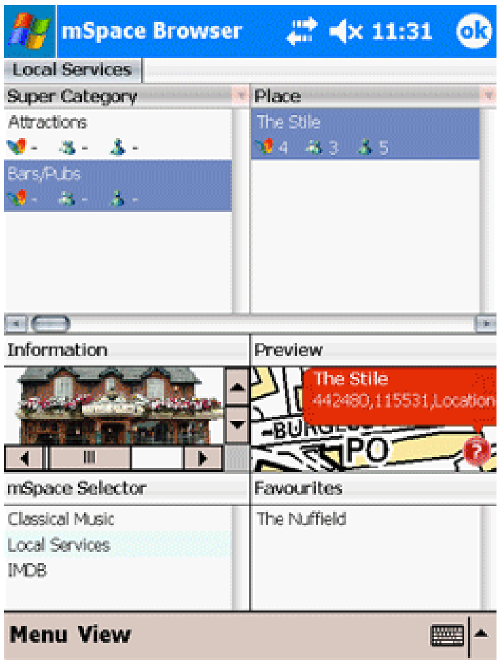
Compound queries are typical when information gathering around requirements when on-the-move; users typically wish to align a set of requirements: where to eat, what to do to kill time, and how to get there. This becomes all the more complex when users preferences are brought into the mix. They may wish to eat a certain kind of cuisine, not want to travel on certain public transport routes (due to disruption or simply preference), and find out if there is a cinema conveniently location along a route. Using web pages to support this kind of query requires the user to jump back and forth between a list of locations and descriptions of locales, as well having to find a place to write down information they wish to remember. This can get unwieldy on the desktop, let alone on a mobile device, as demonstrated by the supporting videos[9] for (Wilson et al., 2006).

When presented with choice, users favour those with trustable recommendations (Smith et al., 2005) and as such, mSpace Mobile allows recommendation and rating by friends and trusted peers. Places that one has previously been to, or near, are also often of interest, and as such a “GPS Ghost” keeps watch of where you go, and subtly highlights return visits. This gives rise to a “recognition”, rather than “recall” activity, a lower cognitive load for the user.

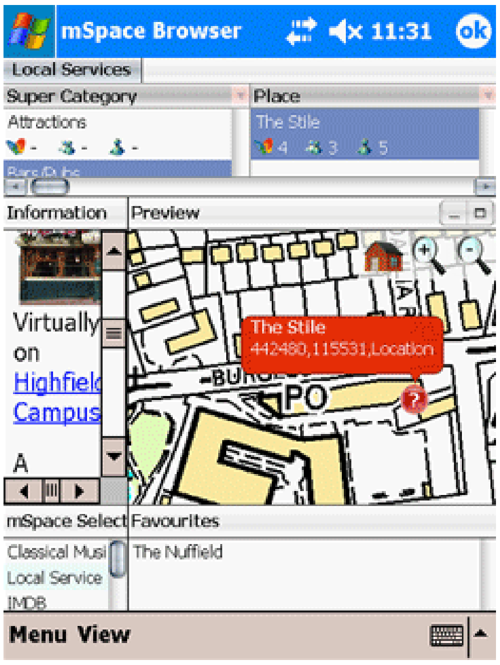
User evaluations showed that such an interface outperformed Google searches, due mostly to the poor performance a small screen web browsing fares (Wilson et al., 2005). Small screen web browsers have to deal with pages that were not designed for small screens, as well as the problem of context shifting when viewing multiple pages, a task that the focus plus context interface of mSpace mobile was showed to perform well at.

mSpace Mobile comprised an implementation of an interface to pure RDF sources, which

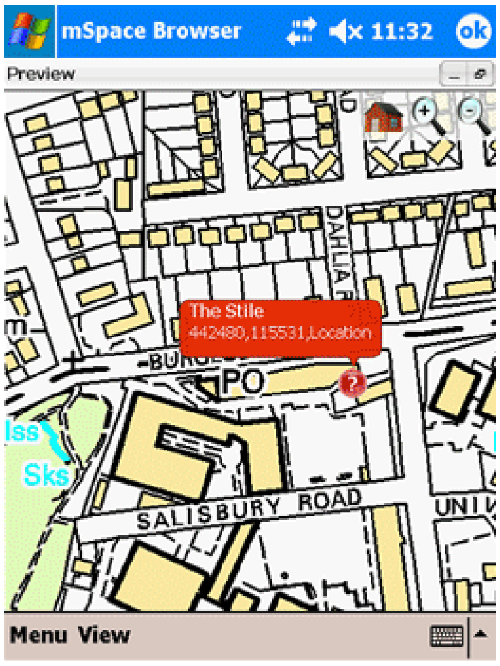
[9]CHI06 supporting videos for mSpace Mobile: `http://eprints.ecs.soton.ac.uk/11886/`



(a) All sections equal.



(b) Map section has partial focus.



(c) Map section has full focus.

FIGURE 5.18: In this example, the user has gone from having equal focus on all sections of the interface, to partially, and then fully focusing on the map section. This is performed by the user double tapping on the section with the stylus, once for partial focus, and the repeating the double tap for full focus.

showed superior interface performance compared to existing systems in user evaluations.

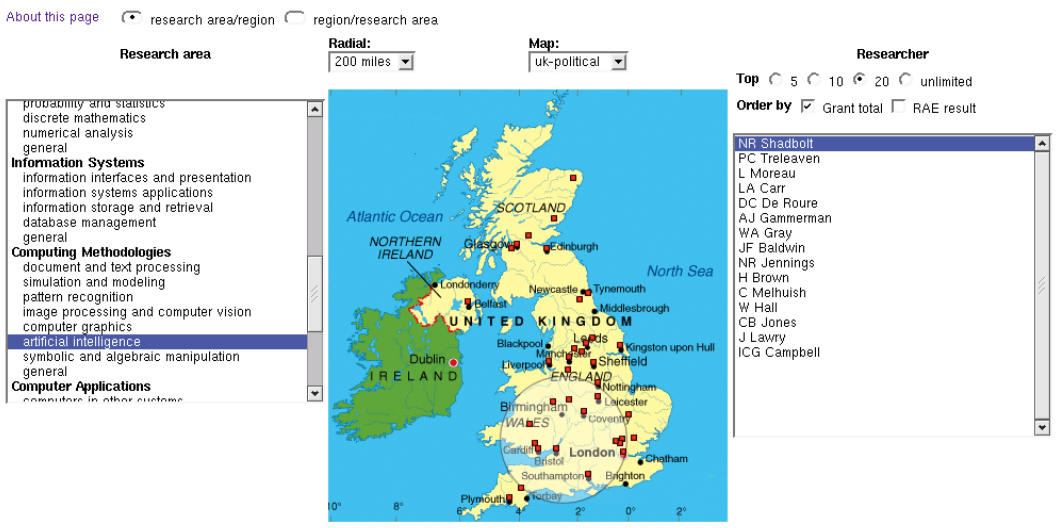
In the next section we expand our investigations into the use of large scale data, and how to tractably deliver performance to the interface when using rich features, such as those that we have developed, which make greater demands upon the query engine.

5.3.3 Scalability Test Cases: Approach 1 - Optimised Triplestore

A series of test cases were performed to determine the technical feasibility of supporting pre-population and backwards highlighting on large data sets. Several potential solutions were examined. The first looks at the 3store triplestore, which had been used to backend previous mSpace work, to investigate its scalability and see if improvements could be made. The second moved on to look at D2R, an alternative solution to the triplestore/database mapping problem. Finally, returning to a standard SQL database was examined, and assessed against the demanding criteria that had been set for the NFO performance.

The first implementations of mSpace utilised the "3store" triplestore software as the storage engine for the displayed metadata. As mSpace was originally an attempt at a generalisable implementation similar to the CSAKTiveSpace interface **??**, the idea to use 3store and the SPARQL query language was taken from this work.

One of the benefits to performance that CSAKTiveSpace utilised is the partitioning of the database that came with the use of a map reticule. The main interface encouraged the use of a reticule over a map of the United Kingdom, as shown in Figure 5.19, that enabled geographical filtering of the results. This type of filtering brings large performance gains, as it effectively partitions the results into localised segments, reducing the processing time and resource requirements of the server and backend software. For example, the user can drag the reticule over Scotland, such that the results might be concerned with Edinburgh and Aberdeen Universities only, or, likewise to the South, such that the results include only those of Southampton and Portsmouth Universities. This partitioning enabled scalability of the system, directed by the interface.


About this page [✓ research area/region] region/research area

Radial: 200 miles

Map: uk-political

Research area

probability and statistics
discrete mathematics
numerical analysis
general
Information Systems
information interfaces and presentation
information systems applications
information storage and retrieval
database management
general
Computing Methodologies
document and text processing
simulation and modeling
pattern recognition
image processing and computer vision
computer graphics
artificial intelligence
symbolic and algebraic manipulation
general
Computer Applications
computers in other systems

Researcher

Top 5 10 [✓ 20] unlimited

Order by [✓ Grant total] RAE result

NR Shadbolt
PC Treleaven
L Moreau
LA Carr
DC De Roure
AJ Gammerman
WA Gray
JF Baldwin
NR Jennings
H Brown
C Melhuish
W Hall
CB Jones
J Lawry
ICG Campbell

FIGURE 5.19: A screenshot of the CSAKTiveSpace interface, showing the map of the UK with reticule for geographic area selection.

Furthermore, by using a triplestore, CSAKTiveSpace benefited from being able to assert distributed resources without having to deal with serialisation or encoding format issues — when data fit the CSAKTiveSpace ontology it appeared in the interface. Thus there remains a desire to utilise a triplestore solution, such as 3store, due to the promise of being able to quickly pull in resources from across the semantic web.

However, the new challenges to the mSpace interface have brought about a need for greater scalability within the storage engine. The inclusion of pre-population, larger datasets and the ability to support more simultaneous users have pushed the limits of performance of 3store. This has meant that the ability to support "querying 40 million triples," as CSAKTiveSpace used, that were previously the *en vogue* measure of storage performance from a triplestore, are now a less useful metric for determining the scalability of realtime delivery of metadata to rich interfaces such as mSpace, due to the increase in the complexity of the requirements.

5.3.4 Scalability Test Cases: Approach 2 - Mapped Database

One of the major implementation restrictions of 3store is its inability to generate compound indexes on a graph beyond single triples, a performance-enhancing feature supported by relational databases for tablular schema data. Moving to a non-semantic relational database engine would have been a major change, especially considering the middle ground offered by D2R Server (Bizer and Cyganiak, 2006). D2R Server allows a SPARQL queryable interface to relational databases, which means that the existing mSpace query generator can be used as-is, while the lower-level storage engine part of the store can be indexed more specifically within the database.

D2R Server is Java-based (Arnold and Gosling, 1998) triplestore middleware software that provides a SPARQL query interface to relational databases. The database tables and columns were mapped to semantic predicates and URIs in the D2RQ language (Bizer and Seaborne, 2004) and D2R Server generates SQL statements and prepares results based on this mapping. This means that we can utilise our existing SPARQL query generation code, while utilising compound multi-column indexes within the underlying database.

Using D2R Server as a question/answer mechanism for generating semantics from existing unsemantic databases is perfectly supported, and works well. For the purpose of highly optimised realtime delivery to the interface, the upgrade to D2R Server was disappointing. While D2R did scale better than 3store on simple queries, some cross-table constraints had the effect of "confusing" the optimiser, in that an overly complex SQL query was generated, leading to unnecessarily poor performance, as the database was fast, but the queries were causing it to perform complex joins that were not required. Another problem was that there is no support for keyword matching, outside of the SPARQL specification(Prud'hommeaux et al., 2005) support of the REGEX (regular expression) operator. This was unsuitable, as the keyword matching was not performed in the database, instead, all of the rows of a table were returned to the D2R server's Java virtual machine, and then a regular expression was matched over them. This process utilised a lot of IO, as well as memory (in both the database and the virtual machine) and as such did not scale well enough for our purposes.

It is likely that in the future is it possible to get D2R working at larger scalability, with bugfixes on the unnecessarily increased complexity problem, as well as non-SPARQL compliant keyword matching support. It is not clear whether the latter of these will ever be supported natively, however D2R is extensible to a certain degree, and as such this may be investigated by a third-party in future, as D2R is open source.

For our work, the interface is the primary driver of the backend scalability. As a user interface researcher, it is important to try to not be constrained by current scalability issues, and to concentrate on innovating features in the interface, and engineering fixes to the middleware if it can't support the interface features sufficiently. It is worth noting that D2R Server is not the only solution to querying of existing databases. ARQ[10] is another similar solution, which shares Jena (Carroll et al., 2003) libraries with D2R.

Following the continued scalability problems with D2R Server, a decision was made to convert the interface back-end code to use MySQL Server, querying using SQL. While this meant that work had to be put in to change the SPARQL creation code to create SQL, the benefits were that full control over the SQL queries were given to the mSpace code, so that optimisations could be made in queries. It also meant that the full features of MySQL Server could be utilised where possible, and that a third-party semantic middleware is not required.

5.3.5 Dynamic List Paging

When the interface tries to enable the exploration of lists that contain very large numbers of items (tens of thousands), sending the text of all of the items to the UI meant sending many megabytes of traffic to the user, potentially at every click. This degraded performance to the point that it was unacceptable, even over broadband connections. It also meant that the processing on the client also took a long time, as the web browser had to process tens of thousands of items. This was mitigated by researching an auto-paged list system, whereby only the applicable **N** items (where **N** is configurable as necessary, we have been typically using 1,000) are sent to the client. This means that if a list would normally result in 55,000 items, only the first 1,000 are sent. When the user scrolls past the downloaded items, an additional request is sent to the server, returning the next items. If the user scrolls quickly further down the list, additional requests are

[10]http://jena.sourceforge.net/ARQ/

sent. Using this method also results in better query times at the storage later, as the queries sent to the database are limited, stopping the query engine processing when it has retrieved enough relevant results from the storage layer. Using dynamic list paging pushed scalability right through to the front-end, so that performance-enhancements were now present in each layer of the system, foregrounding the need for performance as a first-class requirement.

The test case carried out on scalability provided outcomes that can apply to many semantic (and non-semantic) browsing frameworks, and the findings were published (Smith et al., 2007). The outcomes detailed how to query large multi-faceted data, and with multiple ontologies. They show to how to deal with scale, at both query-time and getting data to the UI in a time-efficient and network-efficient way. An additional outcome is that there is now a framework that works at large size to the UI, enabling realtime exploration of large scale data.

5.4 Conclusion

In this chapter we have presented the validations of our work, through formal user studies, tasks now made possible and engineering tractability validation of our work.

We first described our formal user studies that were performed for mSpace Mobile and for our backwards highlighting research. We then outline the tasks now made possible in musicology research through MusicSpace; in bibliographic exploration through Rich Tags and using Facet Ontology to enable pivoting across domains. We ended by describing the engineering tractability of using wild data, knowledge building on-the-go, triplestores and database optimisations and dynamic list paging.

Chapter 6

Conclusion

In this thesis we have discussed the research challenges that were faced when enabling users to create faceted browsing interfaces across heterogeneous data sources, and the contributions that we have presented that address those challenges. The first challenge was how to maintain the integrity over a graph of metadata, when a data graph of metadata from the Semantic Web is applied as a faceted browser.

6.1 Contributions

During this work we have presented the following specific contributions:

- **mSpace Faceted Browser Framework** The mSpace faceted browser framework enables users to browse faceted data through a web browser interface. Developers can assert heterogeneous data from multiple sources into an RDF triplestore, and provide a faceted interface over the data. The framework is scalable and responsive and forms the basis our later work (the contributions of which are described below) where extensions to the framework were made to support simpler setup, user selection of source data, user creation of mSpace interfaces, alignment of data from multiple mSpaces, and pivoting across domains using mSpace.

- **Facet Ontology** In order to inform the faceted interface which data it should populate the faceted interface with, a description of an abstracted view of that data is required. We contribute Facet Ontology, which is used to describe the structure of data that should be use to populate faceted browsers. Facet Ontology is an OWL ontology that allows RDF classes and predicates to be specified as relationships between data that should be put into facets in an interface. Facet Ontology allows transformations of data to be specified, so that when data is harvested, it is filtered through different algorithms before it is rendered and seen by the user. For example, a date facet might be filtered to just a "year," in order to create a Year facet. The use of Facet Ontology means that a model of a domain can be shared, because it is written in RDF, and therefore has a URI, which is resolvable using the methodologies of Linked Data. By sharing descriptions of models, they can be extended and used as a basis for further interfaces of the same, similar or overlapping domains that use the same data source(s).

- **mSpace Maker** To translate a Facet Ontology description into a working faceted browsing interface, a data extraction tool must be used. Our contribution of mSpace Maker performs this task with respect to the creation of faceted interfaces using the mSpace framework. mSpace Maker takes the URI of a Facet Ontology definition, and uses it to extract knowledge from data sources, and populates an mSpace database with that knowledge. A basic mSpace, ready to be customised and styled, is then deployed over that mSpace database. The submitter of the mSpace to the maker is then e-mailed with the URL, and is able to browse the data specified in the Facet Ontology definition. The mSpace Maker is extensible so that additional data transformations can be added, that conform to third-party ontologies.

- **Data Picking Interface** One of the barriers to entry in creating a faceted interface over a data source is being able to explore the source data, and accurately describe the semantic relationships between data. In order to enable all users, and not just those that have experience in the Semantic Web, to be able to pick data from a semantic graph, we contribute the Data Picker. Our tool accepts an entity URI or a SPARQL endpoint URL as its input, and displays data within that graph in minimally technical way. Specifically, we render the data as a scrollable column, as it would look if it were included in a faceted browser. This approach therefore shows users what they are adding to an interface by accepting a particular class of data into their browser. The Data Picker takes a lightweight graph walking approach to picking data, where classes of data available are shown to the user. The user picks a class of interest, and all datatype properties (i.e. all of those that lead to literals) are then rendered as column, and added to the faceted interface. The user can then select any they with to exclude. The default-include nature of the interface is intentional, as it allows users to browse through large scale data sources adding a number of facets without having to select them all individually. When the graph walker encounters any object properties (i.e. predicates that lead to other objects) they are then shown in the graph walker, and all literals of the linked object class are then added to the faceted browser also. The exploration is then a recursive repetition of the above actions, until the user decides they have all of the data they require. The data picker then creates a Facet Ontology definition of the data, and submits this to the mSpace Maker, so that the user can have access to a faceted interface straight away, without the need to submit their definition to the mSpace Maker or handle the definition in any way, if they do not with to.

- **Facet Portal** One of the benefits of our Facet Ontology approach to describing data spaces is that the definition can be shared and built on by others. In order to simplify this process, we contribute the Facet Portal service. This service offers a registry of Facet Ontology definitions, which it indexes so that users can search for interface definitions that match their domain of interest. The portal registry also allows faceted interfaces and data pickers to query the indexes in order to find overlapping domain models, to aid with extending existing models, and pivoting from one model to another.

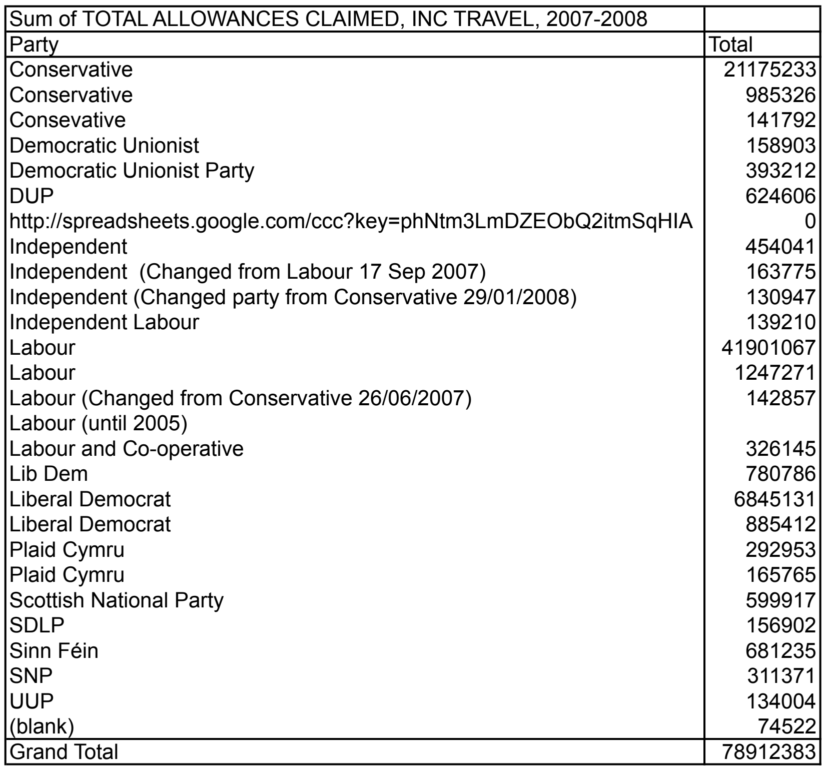
- **Semi-automated Facet Semantic Alignment Service** Our alignment service contributes a lightweight mechanism for semi-supervised co-reference of entities from multiple sources. Entities from data sources described in Facet Ontology definitions are aligned using the Alignment API (Euzenat, 2004) (originally designed for aligning OWL classes, but re-purposed for our faceted alignment). Domain experts can then alter the threshold of alignment, in order to reduce false-positive and false-negative matches from the aligner. The alignments are used to create the pivot between two data sources.

- **Pivoting Faceted Browsing** We contribute pivot support for faceted browsing, so that users of a faceted interface can search for additional domains, and drag in facets from those domains in order to filter their current browsing interface using those facets. Support for pivoting in mSpace provides a lightweight interaction technique that utilises the information on the facets held in a facet portal, so that users can take advantage of facet definitions that exist, and browse across domains, through pivots.

6.2 Future Work

The vision of the Semantic Web as a global unified data source is an ambitious one, requiring both technical scalability and user interface innovation. Through the work described in this thesis, steps have been taken in utilising information from the Semantic Web and using Linked Data that enable responsive browsing of large scale data, across multiple hetergeneous domains. Our work allows users to define their own domains for faceted browsing, and to pivot from definitions of domains that they have defined, to overlapping domains that others have carved out of the Semantic Web. Next steps are to build on this work to enable further breakthroughs in the direction of large-scale unified Semantic Web exploration, which increasingly supports the browsing of wild data that has yet to be curated and described by humans.

Furthermore, innovations in the interactions to perform pivoting can be researched. Specifically, there is scope for further work into the representation of statistical information within a faceted browser. Using the current interface users can explore facets of information, such as "amount claimed" on an MP expenses dataset, however if they wish to ask a question (as opposed to only browsing), such as "Which political party's members claimed the most?" further collation of the data is required. Such a summary can be traditionally achieved through an iteraction known as "pivot tables" — note that this is not the same type of pivot as described in this paper — where information in multiple rows of a spreadsheet is collated, grouped on a chosen column, and totalled. In the MP expenses example, a spreadsheet would have a row for each MP, with their party affiliation and total amount of claims, as available at the Guardian newspapers website: `http://www.guardian.co.uk/news/datablog/2009/may/13/mps-expenses-houseofcommons`. A pivot table would then be created that grouped by affiliation, and totalled the amount claimed, as illustrated in Figure 6.1, which also highlights typical problems in the data, in that duplicates, typos, notes and URIs are present in the "Party" field — highlighting the need for a lightweight mechanism for users to fix errors and pass those fixes back to the source in some secure way.



Sum of TOTAL ALLOWANCES CLAIMED, INC TRAVEL, 2007-2008	
Party	Total
Conservative	21175233
Conservative	985326
Consevative	141792
Democratic Unionist	158903
Democratic Unionist Party	393212
DUP	624606
http://spreadsheets.google.com/ccc?key=phNtm3LmDZEObQ2itmSqHIA	0
Independent	454041
Independent (Changed from Labour 17 Sep 2007)	163775
Independent (Changed party from Conservative 29/01/2008)	130947
Independent Labour	139210
Labour	41901067
Labour	1247271
Labour (Changed from Conservative 26/06/2007)	142857
Labour (until 2005)	
Labour and Co-operative	326145
Lib Dem	780786
Liberal Democrat	6845131
Liberal Democrat	885412
Plaid Cymru	292953
Plaid Cymru	165765
Scottish National Party	599917
SDLP	156902
Sinn Féin	681235
SNP	311371
UUP	134004
(blank)	74522
Grand Total	78912383

FIGURE 6.1: The Pivot Table output of the Guardian's MP Expenses spreadsheet, showing total expense claims, grouped by political party. Data is unedited, and highlights a "dirty data" problem with notes, URIs, duplicates and typos present in the "Party" field. Spreadsheet data as at 8 March 2010.

The kind of summarisation performed by a PivotTable is atypical to the typical exploration-only approaches of faceted browsers, but benefits users as it provides an understandable way to perform calculations (such as summing, counting, averaging and so on) on numerical data, across multiple fields. In addition to PivotTable (used by Microsoft Excel) other work on interfaces to multi-variable aggregation include DataCube (Bosworth et al., 1997), which provides aggregation for each field combination using a novel 3-dimensional cube metaphor.

Further to statistical aggregation, future work can also explore the interface used in the Data Picker (as described in Section 4.5). The current interface is functional and intuitive, benefiting from preventing users from having to select everything, due to its default-include interface. However, in its current design, users do have to explore through object properties manually. Future work can look at how an interface could automate the process of spidering a data graph, so that further facets could be shown immediately, without users having to walk the RDF graph, which we know is something to be avoided (Karger and mc schraefel, 2006). Challenges to this work include:

1. **Determining where to start and what is the first order object**

   The focus of the faceted browser is the class of objects that are chosen as the first order object. All facets relate to this object, and as such the choice of this class alters the meaning of the exploration fundamentally. In the current prototype data picker, the choice of the class is the only major required interaction that the user has to perform. Determining a good place to start, or making the choice more clear to the user is an area that would benefit from further work.

2. **Developing heuristics to determine the depth of auto-exploration**

   In order to prevent the user from having to explore the RDF graph, auto unfolding of the graph, and extracting the data as facets can aid the user in determining all of the possible available information in a data source, so that they can create richer faceted interfaces. However, the challenge in doing so is knowing how deep to explore. In particular, some relationships may cycle back to the original items — *Daniel Smith* tutored by *mc schraefel* who teaches *HCI* which has student *Daniel Smith* — whereas others converge — *JS Bach* composed *Fifth Symphony* which was composed in *1804-1808*. Thus, the former would need a heuristic to say when to stop exploring, whereas the latter stops naturally through its convergence. A heuristic to determine when appropriate tolerances would be beneficial here in aiding users in creating faceted interfaces.

3. **Disregarding duplicate datatypes, or renaming them** The graph structure of Semantic Web data means that cyclical relationships are possible and useful. However, when exploring such data for inclusion into a faceted browser, it may not be useful to include data of the same type in multiple facets. For example, in the DBPedia (Auer et al., 2007) data set, articles are interconnected through multiple predicates, which may not appear in all articles. The article on *The Beatles* links to its *members* Paul McCartney and John Lennon (et al), which then link on, through inverse membership, to other bands that the Beatles members

were in, such as *Wings* and the *Plastic Ono Band*. While this appears reasonable, the additional bands would be then create a duplicate facet, which would be unnecessary. Further work would be beneficial on determining, for known datasets, whether it is acceptable to simply ignore already-included facets (to avoid cycles), or if including them is ever useful, or whether some heuristic can be used to aid users in making this choice.

In addition to helping users create interfaces, future work can progress in aiding to write back the assertions regarding co-reference analysis made by domain experts, in the Facet Alignment Service as described in Section 4.8. Domain experts use the Facet Alignment Service to assert corrections to the automated aligner, and while our service publishes these corrections as RDF, this is not using a standardised mechanism or ontology for asserting changes. Future work to automatically submit these changes (using an appropriate mechanism) back to the data publisher(s), indicating the name, contact details and experience of the domain expert might aid in multiple data sources linking to the resulting nomenclature produced by the assertions of the domain expert. For example, in the Classical Music domain there are many representations of composers, and using the alignment system, a domain expert can produce a set of mappings between sources. By publishing these mappings, third-parties can make use of both sources, and of the hard work performed by the domain expert. If the sources themselves were to link to the mapping, or publish the mappings along with their own data, for example as linked data, both sources would exist online much more closely, which may increase the uptake of both sources by crawlers and other consumers of linked data.

Appendix A

Facet Ontology

A turtle serialisation of the Facet Ontology (`http://danielsmith.eu/resources/facet/#`).

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix : <http://danielsmith.eu/resources/facet/##> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@base <http://danielsmith.eu/resources/facet/#> .

<http://danielsmith.eu/resources/facet/#> rdf:type owl:Ontology .

#################################################################
#
#    Object Properties
#
#################################################################

<http://danielsmith.eu/resources/facet/#class> rdf:type owl:
    ObjectProperty ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#Facet> .

<http://danielsmith.eu/resources/facet/#facetindex> rdf:type owl:
    ObjectProperty ;
  rdfs:range <http://danielsmith.eu/resources/facet/#Facet> ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#
    StandardFacetCollection> .

<http://danielsmith.eu/resources/facet/#faceturi> rdf:type owl:
    ObjectProperty ;
  rdfs:domain [ rdf:type owl:Restriction ;
                owl:onProperty <http://danielsmith.eu/resources/facet/#
    faceturi> ;
                 owl:someValuesFrom <http://danielsmith.eu/resources/facet
    /#StandardFacetCollection>
               ] ;
  rdfs:range [ rdf:type owl:Restriction ;
               owl:onProperty <http://danielsmith.eu/resources/facet/#
    faceturi> ;
               owl:allValuesFrom <http://danielsmith.eu/resources/facet/#
    Facet>
             ] ;
  rdfs:domain [ rdf:type owl:Restriction ;
                owl:onProperty <http://danielsmith.eu/resources/facet/#
    faceturi> ;
                owl:someValuesFrom <http://danielsmith.eu/resources/facet
    /#SliceItem>
             ] .

<http://danielsmith.eu/resources/facet/#labeluri> rdf:type owl:
    ObjectProperty ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#Facet> .

<http://danielsmith.eu/resources/facet/#next> rdf:type owl:ObjectProperty
     ;
  rdfs:range <http://danielsmith.eu/resources/facet/#SliceItem> ;
  rdfs:domain [ rdf:type owl:Restriction ;
                owl:onProperty <http://danielsmith.eu/resources/facet/#
    next> ;
                owl:someValuesFrom <http://danielsmith.eu/resources/facet
    /#Slice>
              ] ,
              [ rdf:type owl:Restriction ;
                owl:onProperty <http://danielsmith.eu/resources/facet/#
    next> ;
                owl:someValuesFrom <http://danielsmith.eu/resources/facet
    /#SliceItem>
              ] .

<http://danielsmith.eu/resources/facet/#slice> rdf:type owl:
    ObjectProperty ;
  rdfs:range <http://danielsmith.eu/resources/facet/#Slice> .

<http://danielsmith.eu/resources/facet/#type> rdf:type owl:ObjectProperty
     ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#Facet> ;
  rdfs:range <http://danielsmith.eu/resources/facet/#FacetType> .

#################################################################
#
#    Data properties
#
#################################################################
```

```
<http://danielsmith.eu/resources/facet/#columngroup> rdf:type owl:
    DatatypeProperty ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#Facet> ;
  rdfs:range rdfs:Literal .

<http://danielsmith.eu/resources/facet/#datasource> rdf:type owl:
    DatatypeProperty ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#Facet> ;
  rdfs:range rdfs:Literal .

<http://danielsmith.eu/resources/facet/#excludefromslice> rdf:type owl:
    DatatypeProperty ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#Facet> .

<http://danielsmith.eu/resources/facet/#generate_summaries> rdf:type owl:
    DatatypeProperty ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#Facet> .

<http://danielsmith.eu/resources/facet/#makertype> rdf:type owl:
    DatatypeProperty ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#
    StandardFacetCollection> ;
  rdfs:range rdfs:Literal .

<http://danielsmith.eu/resources/facet/#maxval> rdf:type owl:
    DatatypeProperty ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#Facet> ;
  rdfs:range rdfs:Literal .

<http://danielsmith.eu/resources/facet/#preprocess> rdf:type owl:
    DatatypeProperty ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#Facet> ;
  rdfs:range rdfs:Literal .

<http://danielsmith.eu/resources/facet/#rdfsource> rdf:type owl:
    DatatypeProperty ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#
    StandardFacetCollection> ;
  rdfs:range rdfs:Literal .

<http://danielsmith.eu/resources/facet/#rdfxmlsource> rdf:type owl:
    DatatypeProperty ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#
    StandardFacetCollection> ;
  rdfs:range rdfs:Literal .

<http://danielsmith.eu/resources/facet/#site> rdf:type owl:
    DatatypeProperty ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#
    StandardFacetCollection> ;
  rdfs:range rdfs:Literal .

<http://danielsmith.eu/resources/facet/#summary_around> rdf:type owl:
    DatatypeProperty ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#Facet> .

<http://danielsmith.eu/resources/facet/#truncatelabel> rdf:type owl:
    DatatypeProperty ;
  rdfs:domain <http://danielsmith.eu/resources/facet/#Facet> .

#################################################################
#
#    Classes
#
#################################################################

<http://danielsmith.eu/resources/facet/#ConnectedFacet> rdf:type owl:
    Class ;
  rdfs:subClassOf <http://danielsmith.eu/resources/facet/#Facet> .

<http://danielsmith.eu/resources/facet/#Facet> rdf:type owl:Class .

<http://danielsmith.eu/resources/facet/#FacetType> rdf:type owl:Class .

<http://danielsmith.eu/resources/facet/#FirstOrderFacet> rdf:type owl:
    Class ;
  rdfs:subClassOf <http://danielsmith.eu/resources/facet/#Facet> .

<http://danielsmith.eu/resources/facet/#Slice> rdf:type owl:Class .

<http://danielsmith.eu/resources/facet/#SliceItem> rdf:type owl:Class .

<http://danielsmith.eu/resources/facet/#StandardFacetCollection> rdf:type
     owl:Class .

#################################################################
#
#    Individuals
#
#################################################################

<http://danielsmith.eu/resources/facet/#TypeLiteral>
  rdf:type
    <http://danielsmith.eu/resources/facet/#FacetType> ,
    owl:NamedIndividual .

<http://danielsmith.eu/resources/facet/#TypeObject>
  rdf:type
    <http://danielsmith.eu/resources/facet/#FacetType> ,
    owl:NamedIndividual .
```

Bibliography

RDF Primer, W3C Technical Reports and Publications. 2004. http://www.w3.org/TR/rdf-primer/, Accessed on 17th July 2006.

H. Alani, C. Brewster, and N. Shadbolt. Ranking ontologies with AKTiveRank. *Lecture Notes in Computer Science*, 4273:1, 2006.

R. Allen. Navigating and searching in hierarchical digital library catalogs. In *Proc. Digital Libraries Conference*, pages 95–100, 1994.

G. Antoniou and F. van Harmelen. Web Ontology Language: OWL. *Handbook on Ontologies*, 2004.

K. Arnold and J. Gosling. *The Java programming language.* ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1998.

Soren Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. chapter DBpedia: A Nucleus for a Web of Open Data, pages 722–735. 2007. `http://dx.doi.org/10.1007/978-3-540-76298-0_52`. 10.1007/978-3-540-76298-0_52.

C. Batini, M. Lenzerini, and SB Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys (CSUR)*, 18(4):364, 1986.

D. Beckett. The design and implementation of the Redland RDF application framework. *Computer Networks*, 39(5):577–588, 2002.

Benjamin B. Bederson, Aaron Clamage, Mary P. Czerwinski, and George G. Robertson. Datelens: A fisheye calendar interface for pdas. *ACM Trans. Comput.-Hum. Interact.*, 11(1):90–119, 2004. ISSN 1073-0516.

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.

T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifiers (URI): generic syntax. 1998.

T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284 (5):28–37, 2001.

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

C. Bizer and A. Seaborne. D2RQ-Treating Non-RDF Databases as Virtual RDF Graphs. *Proceedings of the 3rd International Semantic Web Conference (ISWC2004)*, 2004.

K. Bollacker, R. Cook, and P. Tufts. Freebase: A Shared Database of Structured General Human Knowledge. *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, 22(2):1962, 2007.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.

E.P. Bontas and M. Mochol. Towards a Cost Estimation Model for Ontology Engineering. *Proceedings of the Berliner XML Days Conference*, 2005.

A. Bosworth, J. Gray, A. Layman, and H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Mining and Knowledge Discover*, 1:29–53, 1997.

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

S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 54–68. Springer-Verlag London, UK, 2002.

J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, K. Wilkinson, et al. The Jena Semantic Web Platform: Architecture and design. Technical report, Technical report, Hewlett Packard Laboratories, 2003, 2003.

Trevor Collins, Paul Mulholland, and Zdenek Zdrahal. chapter Semantic Browsing of Digital Collections, pages 127–141. 2005. `http://dx.doi.org/10.1007/11574620_12`. 10.1007/11574620_12.

A. Crossan, R. Murray-Smith, S. Brewster, J. Kelly, and B. Musizza. Gait phase effects in mobile interaction. In *CHI'05 extended abstracts on Human factors in computing systems*, pages 1312–1315. ACM, 2005. ISBN 1595930027.

S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of EMNLP-CoNLL 2007*, pages 708–716, 2007. `http://www.aclweb.org/anthology/D/D07/D07-1074`.

F. Dawson and T. Howes. RFC2426: vCard MIME Directory Profile. *Internet RFCs*, 1998a.

F. Dawson and T. Howes. vCard MIME Directory Profile, 1998b.

S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

M. Dewey. Dewey Decimal Classification System, 1876.

M. Dewey. *Decimal classification.* Forest press, 1951.

P.F. Drucker. Management: tasks, responsibilities, practices. *New York; Harper and Row, 1974, sp Refs*, 1974.

A. Elliott. Flamenco image browser: using metadata to improve image search during architectural design. In *CHI'01 extended abstracts on Human factors in computing systems*, page 70. ACM, 2001.

Orri Erling. Implementing a SPARQL compliant RDF Triple Store using a SQL-ORDBMS. 2006. `http://virtuoso.openlinksw.com/wiki/main/Main/VOSRDFWP`.

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

J. Giles. Internet encyclopaedias go head to head. *Nature*, 438(7070):900–901, 2005.

Hugh Glaser, Tim Lewy, Ian Millard, and Ben Dowling. On coreference and the semantic web, December 2007. `http://eprints.ecs.soton.ac.uk/15245/`.

J. Golbeck and A. Mannes. Using Trust and Provenance for Content Filtering on the Semantic Web. *Proceedings of the Models of Trust for the Web Workshop*, 2006.

T.R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human Computer Studies*, 43(5):907–928, 1995.

Chris Gutteridge. Gnu eprints 2 overview. In *11th Panhellenic Academic Libraries Conference*, 2002.

S. Harris, N. Lamb, and N. Shadbolt. 4store: The Design and Implementation of a Clustered RDF Store. In *The 5th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2009)*, page 86.

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

M. Hearst, A. Elliott, J. English, R. Sinha, K. Swearingen, and K.P. Yee. Finding the flow in web site search. *Communications of the ACM*, 45(9):49, 2002.

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

J. Hendler. On beyond ontology. *Keynote talk, International Semantic Web Conference*, 2003.

Michiel Hildebrand, Jacco van Ossenbruggen, and Lynda Hardman. chapter /facet: A Browser for Heterogeneous Semantic Web Repositories, pages 272–285. 2006. `http://dx.doi.org/10.1007/11926078_20`. 10.1007/11926078_20.

L. Hirschman, P. Robinson, J. Burger, and M. Vilain. Automating coreference: The role of annotated training data. In *Proceedings of the AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, pages 118–121, 1997.

Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: the making of a Web Ontology Language. *J. Web Sem.*, 1(1):7–26, 2003.

David Huynh, Robert Miller, and David Karger. chapter Potluck: Data Mash-Up Tool for Casual Users, pages 239–252. 2007a. `http://dx.doi.org/10.1007/978-3-540-76298-0_18`. 10.1007/978-3-540-76298-0_18.

D.F. Huynh, D.R. Karger, and R.C. Miller. Exhibit: lightweight structured data publishing. *Proceedings of the 16th international conference on World Wide Web*, pages 737–746, 2007b.

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

Afraz Jaffri, Hugh Glaser, and Ian Millard. An infrastructure for managing uri synonymity on the semantic web (poster). In *5th European Semantic Web Conference*. Springer, June 2008. `http://eprints.ecs.soton.ac.uk/15677/`.

K.S. Jones et al. A Statistical Interpretation of Term Specificity and its Application in Retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

David Karger and mc schraefel. The pathetic fallacy of rdf. In *International Workshop on the Semantic Web and User Interaction (SWUI) 2006*, 2006. `http://eprints.ecs.soton.ac.uk/12911/`.

K.J. Klauer. Intentional and incidental learning with instructional texts: A meta-analysis for 1970–1980. *American educational research journal*, 21(2):323, 1984. ISSN 0002-8312.

R. Lee. Scalability Report on Triple Store Applications. *Massachusetts institute of technology*, 2004.

C. Lynch, S. Parastatidis, N. Jacobs, H. Van de Sompel, and C. Lagoze. The OAI-ORE effort: progress, challenges, synergies. *Proceedings of the 2007 conference on Digital libraries*, pages 80–80, 2007.

m. c. schraefel, Maria Karam, and Shengdong Zhao. mspace: interaction design for user-determined, adaptable domain exploration in hypermedia. In Paul De Bra, editor, *AH 2003: Workshop on Adaptive Hypermedia and Adaptive Web Based Systems*, pages 217–235, 2003.

m. c. schraefel, Daniel A. Smith, Alisdair Owens, Alistair Russell, Craig Harris, and Max Wilson. The evolving mspace platform: leveraging the semantic web on the trail of the memex. In *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, pages 174–183, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-168-6.

B. MacKay, D. Dearman, K. Inkpen, and C. Watters. Walk'n scroll: a comparison of software-based navigation techniques for different levels of mobility. In *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 183–190. ACM, 2005. ISBN 1595930892.

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.

A. Mannes. *Profiles In Terror: The Guide To Middle East Terrorist Organizations.* Rowman & Littlefield Publishers, 2004.

A. Mannes and J. Golbeck. Building a Terrorism Ontology. *ISWC Workshop on Ontology Patterns for the Semantic Web*, 2005. `http://www.mindswap.org/papers/TerrorOntologyfinal.pdf`.

C.C. Marshall and F.M. Shipman III. Spatial hypertext and the practice of information triage. *Proceedings of the eighth ACM conference on Hypertext*, pages 124–133, 1997.

m.c. schraefel. What is an analogue for the semantic web and why is having one important? In *ACM Hypertext 2007*, 2007. `http://eprints.ecs.soton.ac.uk/14274/`. ACM Englebart Best Paper Award.

m.c. schraefel, Maria Karam, and Shengdong Zhao. Audio preview cues: Interaction aides for exploration of online music and beyond. In *HCI International 2003*, 2003.

http://eprints.ecs.soton.ac.uk/8801/. The first PDF is to the paper version of the poster published in the proceedings; the second pdf is a copy of the actual poster presented at the conference.

mc schraefel, Nigel R. Shadbolt, Nicholas Gibbins, Stephen Harris, and Hugh 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. ISBN 1-58113-844-X.

m.c. schraefel, Max Wilson, Alistair Russell, and Daniel A. Smith. mspace: improving information access to multimedia domains with multimodal exploratory search. *Commun. ACM*, 49(4):47–49, 2006. ISSN 0001-0782.

Michael J. McGuffin and m. c. schraefel. A comparison of hyperstructures: zzstructures, mspaces, and polyarchies. In *HYPERTEXT '04: Proceedings of the fifteenth ACM conference on Hypertext and hypermedia*, pages 153–162, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-848-2.

Duncan McRae-Spencer and Nigel 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. http://eprints.ecs.soton.ac.uk/12704/.

G.A. Miller. WordNet: A Lexical Database for English. *COMMUNICATIONS OF THE ACM*, 38(11):39, 1995.

W. Mischo. Library of Congress Subject Headings. *Cataloging & Classification Quarterly*, 1(2):105–124, 1982.

S. Mizobuchi, M. Chignell, and D. Newton. Mobile text entry: relationship between walking speed and text input task difficulty. In *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 122–128. ACM, 2005. ISBN 1595930892.

BA Myers. Visual programming, programming by example, and program visualization: a taxonomy. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 59–66, 1986.

AB MySQL. MySQL, 2004.

Michael L. Nelson, Herbert Van de Sompel, and Carl Lagoze. Report on the 2nd workshop on the open archives initiative, gaining independence with e-print archives and oai: 17 - 19 october 2002, cern, switzerland. *D-Lib Magazine*, 8(11), 2002.

J. Nielsen. *Usability engineering.* Morgan Kaufmann, 1993.

J. Nielsen. The need for speed. *Alertbox (web page: http://www.useit.com/alertbox/9703a.html)*, 1997.

I. Niles and A. Pease. Towards a standard upper ontology. *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, pages 2–9, 2001.

C. Ondrejka. Escaping the gilded cage: User created content and building the metaverse. *NYL Sch. L. Rev.*, 49:81, 2004.

Tim O'Reilly. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. *Communications & Strategies*, 1:17, 2007.

Eyal Oren, Renaud Delbru, and Stefan Decker. chapter Extending Faceted Navigation for RDF Data, pages 559–572. 2006. `http://dx.doi.org/10.1007/11926078_40`. 10.1007/11926078_40.

A. Passant and P. Laublet. Meaning Of A Tag: A collaborative approach to bridge the gap between tagging and Linked Data. In *Proceedings of the WWW 2008 Workshop Linked Data on the Web (LDOW2008), Beijing, China*, 2008.

S. Payette and C. Lagoze. flexible and Extensible Digital Object and Repository Architecture (FEDORA). *Research and Advanced Technology for Digital Libraries: Second European Conference, ECDL'98, Heraklion, Crete, Cyprus, September 21-23, 1998: Proceedings*, 1998.

Bill Pere. *Songcrafters' Coloring Book: The Essential Guide to Effective and Successful Songwriting.* Connecticut Songwriting Academy Press, 2009. ISBN 978-0977770809.

Emmanuel Pietriga, Christian Bizer, David Karger, and Ryan Lee. chapter Fresnel: A Browser-Independent Presentation Vocabulary for RDF, pages 158–171. 2006. `http://dx.doi.org/10.1007/11926078_12`. 10.1007/11926078_12.

D. Pitzalis, C. Lahanier, R. Pillay, G. Aitken, A. Russell, D. A. Smith, P. A. S. Sinclair, M. J. Addis, R. Lowe, S. Hafeez, P. H. Lewis, K. Martinez, and m. c. schraefel. Semantically exposing existing knowledge repositories: a case study in cultural heritage. In *The first international conference on Semantics And digital Media Technology (SAMT 2006)*, 2006. `http://eprints.ecs.soton.ac.uk/14270/`.

E. Prud'hommeaux, A. Seaborne, et al. SPARQL Query Language for RDF. *W3C Working Draft*, 23, 2005.

D. Quan, D. Huynh, and D.R. Karger. Haystack: A Platform for Authoring End User Semantic Web Applications. *International Semantic Web Conference*, pages 738–753, 2003.

D. A. Quan and R. Karger. How to make a semantic web browser. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 255–265, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-844-X.

Y. Raimond, S. Abdallah, M. Sandler, and F. Giasson. The music ontology. In *Proceedings of the International Conference on Music Information Retrieval*, pages 417–422, 2007.

Ramana Rao and Stuart K. Card. The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 318–322, New York, NY, USA, 1994. ACM Press. ISBN 0-89791-650-6.

G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins. Polyarchy visualization: visualizing multiple intersecting hierarchies. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, pages 423–430. ACM New York, NY, USA, 2002a.

George Robertson, Kim Cameron, Mary Czerwinski, and Daniel Robbins. Polyarchy visualization: visualizing multiple intersecting hierarchies. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 423–430, New York, NY, USA, 2002b. ACM Press. ISBN 1-58113-453-3.

Jack Rusher. Rhetorical Device: Triple Store. 2001. http://www.w3.org/2001/sw/Europe/events/20031113-storage/positions/rusher.html, Accessed on 17th July 2006.

R. Rustin. *Natural language processing.* Algorithmics Press, 1973.

B. Schwartz. *The paradox of choice: Why more is less.* Harper Perennial, 2005.

A. Sears and B. Shneiderman. Split menus: effectively using selection frequency to organize menus. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 1(1): 27–51, 1994. ISSN 1073-0516.

A.J. Sellen, R. Murphy, and K.L. Shaw. How knowledge workers use the web. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, pages 227–234. ACM, 2002.

D Smith, S Menon, and K Sivakumar. Online peer and editorial recommendations, trust, and choice in virtual markets. *Journal of Interactive Marketing*, 19(3):15–37, 2005.

Daniel Alexander Smith. Ontobrowse: A world of knowledge. `http://eprints.ecs.soton.ac.uk/13802/`. This is the final report for the Part III Project., May 2004.

Daniel Alexander Smith, David Bretherton, mc schraefel, Richard Polfreman, Mark Everist, Jeanice Brooks, and Joe Lambert. Using pivots to explore heterogeneous collections: A case study in musicology. In *All Hands Meeting 2009*, 2009. `http://eprints.ecs.soton.ac.uk/17998/`.

Daniel Alexander Smith, Joe Lambert, and m c schraefel. Rich tags: Cross-repository browsing. In *Open Repositories Conference 2008 (OR2008)*, 2008. `http://eprints.ecs.soton.ac.uk/15130/`.

Daniel Alexander Smith, Alisdair Owens, mc schraefel, Patrick Sinclair, Paul André, Max Wilson, Alistair Russell, Kirk Martinez, and Paul Lewis. Challenges in supporting faceted semantic browsing of multimedia collections. In *The Second International Conference on Semantic and Digital Media Technologies (SAMT2007)*. Springer, 2007. `http://eprints.ecs.soton.ac.uk/14507/`.

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

M. Streatfield and H. Glaser. Report on Summer Internship Work For the AKT Project: Benchmarking RDF Triplestores. 2005.

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

R. Tansley, M. Bass, D. Stuve, M. Branschofsky, D. Chudnov, G. McClellan, and M. Smith. The DSpace institutional digital repository system: current functionality. *Digital Libraries, 2003. Proceedings. 2003 Joint Conference on*, pages 87–97, 2003.

R.J. Todd. From information to knowledge: charting and measuring changes in students' knowledge of a curriculum topic. *Information Research*, 11(4):11–4, 2006.

G. Tummarello, R. Delbru, and E. Oren. Sindice. com: Weaving the open linked data. *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007+ ASWC 2007, Busan, Korea, November 11-15, 2007*, 4825:552, 2007.

H. Van de Sompel, M.L. Nelson, C. Lagoze, and S. Warner. Resource Harvesting within the OAI-PMH Framework. *D-Lib Magazine*, 10(12):1082–9873, 2004.

J. Voss. Tagging, folksonomy & co-renaissance of manual indexing? *Arxiv preprint cs/0701072*, 2007.

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.

S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. RFC2413: Dublin Core Metadata for Resource Discovery. *Internet RFCs*, 1998.

Jan Wielemaker, Michiel Hildebrand, Jacco van Ossenbruggen, and Guus Schreiber. chapter Thesaurus-Based Search in Large Heterogeneous Collections, pages 695–708. 2008. `http://dx.doi.org/10.1007/978-3-540-88564-1_44`. 10.1007/978-3-540-88564-1_44.

Max Wilson, Alistair Russell, mc schraefel, and Daniel A. Smith. mspace mobile: a ui gestalt to support on-the-go info-interaction. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 247–250, New York, NY, USA, 2006. ACM Press.

Max Wilson, Alistair Russell, Daniel A. Smith, Alisdair Owens, and m.c. schraefel. mspace mobile: A mobile application for the semantic web. In *4th International Semantic Web Conference*, 2005.

Max L Wilson, m.c. schraefel, and Ryen W. White. Evaluating advanced search interfaces using established information-seeking models. *In the Journal of the American Society for Information Science and Technology*, July 2009. `http://eprints.ecs.soton.ac.uk/14301/`.

Ching Man Au Yeung, Nicholas Gibbins, and Nigel Shadbolt. Understanding the semantics of ambiguous tags in folksonomies. In *The International Workshop on Emergent Semantics and Ontology Evolution (ESOE2007) at ISWC/ASWC 2007*, November 2007. `http://eprints.ecs.soton.ac.uk/14869/`.