

University of Southampton Research Repository ePrints Soton

Copyright © and Moral Rights for this thesis are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given e.g.

AUTHOR (year of submission) "Full thesis title", University of Southampton, name of the University School or Department, PhD Thesis, pagination

UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL AND APPLIED SCIENCES

Electronics and Computer Science

End-user Data-centric Interactions over Linked Data

by

Igor O. Popov

Thesis for the degree of Doctor of Philosophy

July 2013

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL AND APPLIED SCIENCES

Electronics and Computer Science

Doctor of Philosophy

END-USER DATA-CENTRIC INTERACTIONS OVER LINKED DATA

by **Igor O. Popov**

The ability to build tools that support gathering and querying information from distributed sources on the Web rests on the availability of structured data. Linked Data, as a way for publishing and linking distributed structured data sources on the Web, provides an opportunity to create this kind of tools. Currently, however, the ability to complete such tasks over Linked Data sources is limited to users with advanced technical skills, resulting in an online information space largely inaccessible to non-technical end users. This thesis explores the challenges of designing user interfaces for end users, those without technical skills, to use Linked Data to solve information tasks that require combining information from multiple sources. The thesis explores the design space around interfaces that support access to Linked Data on demand, suggests potential use cases and stakeholders, and proposes several direct manipulation tools for end users with diverse needs and skills. User studies indicate that the tools built offer solutions to various challenges in accessing Linked Data that are identified in this thesis.

Contents

Declaration of Authorship	xv
1 Introduction	1
1.1 Problem Definition and Scope	3
1.2 Research Challenges	5
1.3 Approach	9
1.3.1 Deriving Requirements for End-user Data-centric Tools	9
1.3.2 Visor: Exploring Unknown Data Domains	11
1.3.3 mashpoint: Linking Data-centric Web Applications	13
1.3.4 Scope and Limitations	14
1.4 Contributions	14
1.4.1 Publications relating to thesis work	15
1.4.2 Other Publications	17
1.5 Thesis Overview	18
2 Background and Related Work	21
2.1 Structured Data	21
2.1.1 Uses of Structured Data in Information Exploration	22
2.1.1.1 Organisation of Information Content	23
2.1.1.2 Faceted Browsing and Filtering	24
2.1.1.3 Visualisations and Multiple Representations	25
2.1.1.4 Data-centric operations	26
2.2 Structured Data on the Web	27
2.2.1 Sources of Structured Data	27
2.2.1.1 Use of Data Technologies on the Web	28
2.2.1.2 User Contributed Content	29
2.2.1.3 Structuring Unstructured Data	30
2.2.1.4 Page Markup	30
2.2.2 The Semantic Web	31
2.2.2.1 The Web of Linked Data	32
2.2.2.2 Concepts	33
2.3 Data-centric Interfaces over Linked Data	35
2.3.1 Defining Data-centric Interfaces for End Users	36
2.3.2 The Case for Supporting Data-centric Interactions over Linked Data	36
2.3.3 Challenges of Developing Data-centric Tools for Exploring Linked Data	37
2.3.3.1 Front-end Challenges	37

2.3.3.2	Back-end Challenges	38
2.4	Areas Related to End-user Data-centric Interfaces	40
2.4.1	Visual Query Languages	40
2.4.2	End User Mashup Tools	41
2.5	User Interfaces over Linked Data	43
2.5.1	Representing Data	44
2.5.1.1	Direct Visualisation Approaches	44
2.5.1.2	Use of Lenses and Templates	45
2.5.2	Data-mapping Interfaces	47
2.5.2.1	Faceted Browsers over Linked Data	47
2.5.2.2	Widget Library Approaches	48
2.5.3	Data-Browsers and Navigational Interfaces	49
2.5.4	Mashup Editors	52
2.5.5	Keyword Search and Natural-Language Processing	52
2.6	Summary	53
3	Design Process for Data-centric Interactions over Linked Data	55
3.1	Design Process Goals	56
3.2	Method	57
3.3	Personas for Data-centric Interactions	59
3.3.1	Walkthrough	61
3.3.2	Task analysis	64
3.3.2.1	Stakeholders	66
3.3.3	Linked Data Scenario	66
3.3.3.1	Linked Data Publishing Practices	67
3.3.3.2	Anatomy of a Linked Data Generic Browser	68
3.3.3.3	Available Linked Data	69
3.3.3.4	Challenges in Interacting over Linked Data	70
3.4	Analysis of Interfaces Supporting Data-centric Interactions over Linked Data	72
3.4.1	Exploration Starting Point	74
3.4.2	Navigation and Browsing	75
3.4.3	Data Representation	79
3.4.3.1	Representation of individual resources	79
3.4.3.2	Representation of sets	81
3.4.3.3	Visual aids and tools for analytics	83
3.4.4	Query and Filtering	83
3.4.4.1	Filtering across trails of navigation	83
3.4.4.2	Additional filtering options	85
3.4.4.3	User Studies and Evaluation	85
3.4.5	Observations and Conclusions	85
3.5	Prototyping a Generic Data Browser - The GEORDi Experience	87
3.5.1	Design Goals	88
3.5.2	System Description	88
3.5.2.1	Exploration Starting Point	88
3.5.2.2	Data Representation and Browsing	90
3.5.2.3	Tools for Visualisations and Analysis	92

3.5.3	Deployment and Observations	93
3.6	Summary	94
4	Multi-pivot Exploration of Data on the Web	97
4.1	Limitations of Navigation-based Browsers	98
4.2	Approach	101
4.3	User Interface	102
4.3.1	Data and Ontology Exploration	102
4.3.1.1	Collection inspector	105
4.3.1.2	Instance inspector	106
4.3.1.3	Relations inspector.	106
4.3.2	Spreadsheet Creation	106
4.3.2.1	Main collection.	107
4.3.2.2	Adding columns.	107
4.3.2.3	Defining column paths.	107
4.3.3	Implementation	109
4.4	User Study	110
4.4.1	Study Design and Procedure	110
4.4.2	Results	111
4.4.3	Implications for Design	113
4.5	Summary	114
5	mashpoint - Browsing the Web along Structured Lines	117
5.1	mashpoint - Concepts and Design	118
5.1.1	User Experience	119
5.1.2	Interaction Concepts	123
5.1.2.1	Homogeneous Collections between Applications	123
5.1.2.2	Different Information-concept Collections between Applications	125
5.1.3	Defining mashpoint	126
5.1.3.1	Lens Perspective	126
5.1.3.2	Application Pipelining Perspective	128
5.1.3.3	Structured Data Clipboard	128
5.2	The Case for Application Level Abstractions	128
5.2.1	Representation Problem	129
5.2.2	Dealing with Fine-grained Raw Data	129
5.2.3	Data Overload	129
5.2.4	Social Contribution Factor	130
5.3	Implementation	130
5.3.1	Applications	131
5.3.2	mashpoint Discovery Service	133
5.3.3	Pivoting Across Applications	133
5.4	Evaluation	135
5.4.1	Data and Application Gathering	135
5.4.2	First User Study	136
5.4.2.1	Study Design and Procedure	136
5.4.2.2	Results	137

5.4.3	Second User Study	140
5.4.3.1	Study Design and Procedure	140
5.4.3.2	Results	142
5.5	Enriching Unstructured Content in Websites	143
5.6	Summary	144
6	Discussion: Implications, Challenges and Future Directions	145
6.1	Generic Data Browsers	147
6.1.1	Establishing Standard Ontologies or Vocabularies	147
6.1.2	End Users and Data Heterogeneity	147
6.2	Exploring Unfamiliar Datasets	148
6.3	mashpoint	149
6.3.1	The Co-reference Problem and Approximate Semantics	150
6.3.2	Cost/Benefit of Using Linked Data Technologies	152
6.3.2.1	Low Barrier for Entry	152
6.3.2.2	Incentives for Publishing and Linking Applications	152
7	Conclusions and Future Work	155
7.1	Summary of Contributions	155
7.2	Future Work	156
7.2.1	Collaboration	156
7.2.2	Crowdsourcing Data-oriented Tasks	157
7.2.3	Supporting Domain-specific Areas	157
	References	159

List of Figures

1.1	A Google search result page showing results for a query attempting to find a web page showing GDP per capita alongside population data for countries.	2
1.2	Scope of thesis from the whole set of challenges of delivering data browsers over Linked Data.	4
1.3	The GEORDi generic data browsers showing a spreadsheet generated from Linked Data and a graph widget to visualise the data.	11
1.4	The Visor browser offering an overview-first, instance data on demand way of browsing a Linked Dataset.	12
1.5	Browsing for one data-centric Web page to another with a set of entities.	13
1.6	Thesis chapter organisation based on work conducted.	16
2.1	A typical online shopping site offering data-centric features to filter for information.	22
2.2	The Flamenco faceted browser.	25
2.3	The Spotfire tool using different visualisations over the same source of structured data (Tanin et al. (1997)).	26
2.4	Different forms and uses of structured data on the Web.	28
2.5	A Google search on recipes shows rich visual feedback due to structured data imbedded in web sites.	31
2.6	The Semantic Web technology layer cake.	32
2.7	The Linked Open Data cloud shows datasets publicly available on the Web and the links they have to other datasets.	33
2.8	Information about the book “Lord of the Rings” in RDF.	34
2.9	The Visage tool integrating visualisations with a visual query language (Derthick et al. (1997)).	41
2.10	The WebSummaries tool Dontcheva et al. (2006).	42
2.11	The YahooPipes user interface.	43
2.12	Use of lenses to display RDF data. Lens can be deployed per resource or can be used to show aggregate views of multiple RDF resources.	46
2.13	The mSpace column faceted browser supporting backward highlighting. .	48
2.14	A pivoting example showing a pivot from a collection of places to rentals.	50
2.15	The Tabulator interface for exploring Linked Data. Each navigated resource is displayed as a nested concept of the resource from which the navigation originated.	51
2.16	The Parallax browser. It allows set-oriented interaction with multiple resources. It supports faceted filtering of each collection (left) and a list of connection to which the collection can pivot (upper right).	52

3.1	The Scenario-based design process proposed by Rosson and Carroll (2003).	58
3.2	The Disco Generic Data Browser.	70
3.3	A conceptual ontology for data about UK Members of Parliament.	71
3.4	Pivoting with sets of resources.	77
3.5	Tabulator, Humboldt and gFacet showing different data browsing visualisations	77
3.6	Example concept of branching in a generic data browser.	78
3.7	Parallax showing multiple navigational trails displayed in a spreadsheet. .	83
3.8	Visual aids offered in Tabulator and Parallax to visualize and analyse collected data.	84
3.9	The GEORDi catalogue of datasets.	89
3.10	Collections contained within a dataset represented in GEORDi.	89
3.11	Concept of an irregular table to visualise graphs of data. The Figure shows entities described in cells. The red relationships illustrates how the entities are linked; the type of relationship is indicated in the column header.	91
3.12	An example showing set-oriented operations in GEORDi over statistical data of UK government linked datasets. Users can open up new columns from any existing column.	91
3.13	Filtering in GEORDi. The filters allow users to filter through the unique values of each of the columns.	92
3.14	The final GEORDi prototype showing tools to visualise and analyse collected data.	93
4.1	A portion of the DBPedia ontology showing the links between instances of the classes Olympic Games, Countries, Cities and Person needed to answer a query.	99
4.2	A graph visualisation of the DBPedia dataset ontology showing which classes are linked. Each node represents a class and each arc means that there is at least one property connecting instances between two classes. The Figure highlights the classes described in Figure 4.1.	100
4.3	Generating a subset of the DBPedia ontology generated by selecting concepts in the ontology in Visor. Selected concepts (e.g. “Olympic Games”) are coloured in green, while suggested concepts are coloured in grey. The arcs between the two collections with a blue node in the middle indicate links between items (e.g. “Olympic Games has direct relationship with “Cities.	103
4.4	Inspectors showing diverse information on the concepts in the ontology. Information about “Countries is shown in (a), and information about a particular country (in this case Singapore) is shown in (b). The various links that exist between “Countries and “Cities is shown in (c).	104
4.5	Displaying properties in the collection inspector that link from or to items of a collection.	105
4.6	Spreadsheet creating/query interface in Visor. Users start by selecting the main collection (a) datatype properties (b) columns from other collections in the sub-ontology by specifying relations to the main collection (c) and (d). A preview of the columns is shown in (e).	108
4.7	Specifying a path from the main collection to an arbitrary collection in Visor.	109

4.8	Reliance in showing instance level data as opposed to collection overviews in Visor.	112
5.1	An example of pivoting with data between two applications in mashpoint.	120
5.2	Example applications linked with the mashpoint framework.	121
5.3	Displaying different types of information in the mashpoint application selector window.	124
5.4	Navigation between applications which support entities from different information concept types.	125
5.5	Applications in mashpoint viewed as lenses over a graph of data.	127
5.6	Preserving the state in a mashpoint-linked application.	132
5.7	Architecture of the mashpoint framework.	134
5.8	Ranking the difficulty of tasks in the exploratory user study.	137
5.9	Mashpoint applications embedded in a browser-like application and extended with faceted search.	141
5.10	Average time it took to complete the tasks with Navigator Interface versus the Browsing interface.	142
5.11	Evaluating the difficulty of mashing data using multiple applications in the mashpoint framework between the two groups using a Likert scale. The Figure shows two how users from both groups ("Filter" and "Search") rated the difficulty of accomplishing their task on a scale of 1-9 (1 - Very Easy, 9 - Very Hard).	143
5.12	Using mashpoint to show structured information about entities in unstructured pages.	144
6.1	Landscape of data-centric tools over Linked Data.	146
6.2	Data transformation from two sources in OpenRefine using macros. The Figure shows transforming numeric data by aligning different representations of rounding.	149
6.3	The Google Squared tool. After a search results in a list of entities, users can add columns by typing in column names, which are used as a keyword to search for those properties in the initial list of entities.	150
6.4	Facebook's Graph Search.	151
6.5	Google's Knowledge Graph showing structured information about planets.	151

List of Tables

3.1	Exploration starting point implementations across different browsers. . . .	75
3.2	Browsing methods supported in different browsers.	76
3.3	Data representation across browsers.	80
3.4	Evaluation and user studies made on Linked Data browsers.	85
5.1	Training and evaluation tasks used in the user study. Column 2 shows the type of operations involved in completing each task.	138

Declaration of Authorship

I, **Igor O. Popov** , declare that the thesis entitled *End-user Data-centric Interactions over Linked Data* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly while in candidature for a research degree at this University;
- 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;
- where I have consulted the published work of others, this is always clearly attributed;
- 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;
- I have acknowledged all main sources of help;
- 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;
- none of this work has been published before submission except where stated in the text

Signed:.....

Date:.....

Chapter 1

Introduction

The World Wide Web (WWW) is arguably the single most used information medium for answering our information needs. Whenever we need to access information on the Web, we usually turn to Web search engines, which enable us to find relevant Web pages through a simple keyword search interface. Keyword search is undoubtedly the prevalent model to support various forms of information-seeking activities on the Web, from simple fact-finding activities such as "look up the weather forecast" to more complex information-gathering activities such as "finding the best apartment to rent". While search algorithms that rank documents are continuously improved, the biggest factor of success in a keyword search query still lies in the existence of a document on the Web that contains all the relevant information we require to complete the information task. In other words, if the information we require for a particular need is scattered across several pages, then search engines can, at best, only surface the individual Web pages containing parts of the relevant information. In such situations we are left to manually perform the task of gathering, structuring and combining the relevant information.

To illustrate these challenges consider the following examples. For simple information needs, for example, if one needs to get information about the GDP per capita for the UK, one simply needs to look up this information in the corresponding Wikipedia article for that country¹. Similarly, if one wants to compare the UK to other countries, one can find this aggregate information on a specially dedicated Wikipedia page², which lists data about countries in a table and allows users to sort columns on the topic. However, if one needs to find the geographic distribution of the bottom 25% of countries according to GDP per capita that have a population of 10-20M, then one would find it difficult to find a single Web page containing exactly that fact. Even a more modest information request, for example, finding a page that has the information of GDP per capita alongside the population data, would prove difficult to find (Figure 1.1). As the

¹http://en.wikipedia.org/wiki/United_Kingdom

²[http://en.wikipedia.org/wiki/List_of_countries_by_GDP_\(PPP\)_per_capita](http://en.wikipedia.org/wiki/List_of_countries_by_GDP_(PPP)_per_capita)

search results³ displayed in Figure 1.1 show, a keyword search would only yield various Web documents that are relevant to the query, however no single page containing all the relevant data in one place.

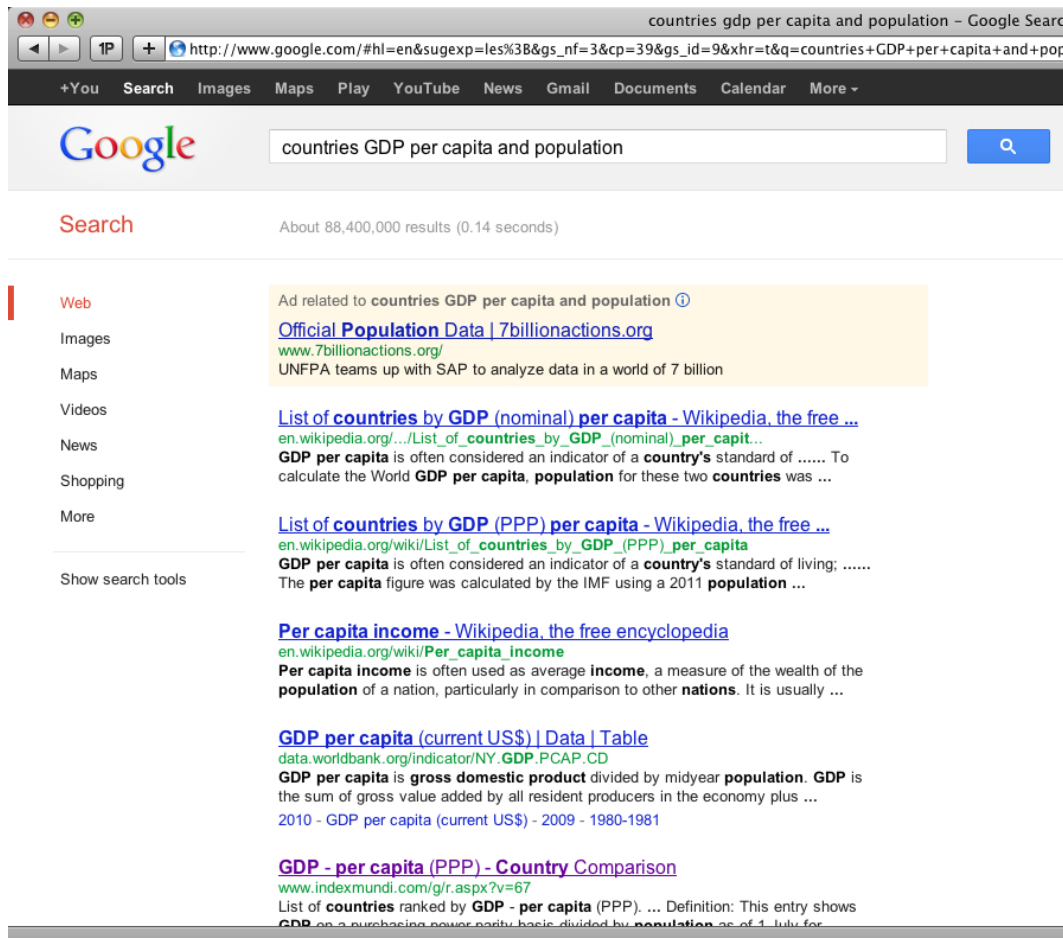


Figure 1.1: A Google search result page showing results for a query attempting to find a web page showing GDP per capita alongside population data for countries.

Thus, to complete the task we would have to spend a significant time doing data-related activities. In a hypothetical scenario of being given such an information task, we would need to first select the bottom 25% low-income countries from the Wikipedia page on GDP per capita data, and then store this information into a tool such as a spreadsheet or a note-taking tool for further processing. What follows is the tedious task of looking up other countries population in the second page in order to filter out those that do not have a population size of 10-20M. Once the final list of countries is compiled, we would need to look up geographic information, or alternatively use a mapping tool, preferably one that allows us to make an input of the set of countries into an aggregate, in order to mitigate the need of individually looking up the location of each country. As the scenario suggests, the cost of finding the answer to these types of *data-centric* queries

³The query was performed on the Google search engine, November 2012

is significantly higher in the absence of a single web page containing all of the required information.

The reason why the Web falls short when dealing with this sort of *data-centric* queries is due to the fact that the Web is largely comprised of unstructured, text-based documents. Therefore, at present, the best available method to support data-centric interactions is to manually do the task of extracting, combining, and integrating data gathered from multiple websites.

The solution to providing more efficient data-centric interactions begins with the availability of structured data on the Web. Today the Web begins to include access to structured data in various forms - from RSS feeds, to data APIs, and data dumps i.e. downloadable documents of structured data. While publishing structured data is an important first step, it still does not provide a solution to the problem of integrating multiple data sources on demand. *Linked Data*⁴ is an effort to publish structured data on the Web that addresses the challenges of data integration (Heath and Bizer (2011)). Once the data is published as Linked Data, links can be established between the entities of different sources of Linked Data, much like documents can be linked on the Web. By establishing links between entities from different datasets, the Linked Data effort essentially aims at creating a Web of Data alongside the established Web of Documents.

Linked Data provides an infrastructure for accessing data on the Web. At present, however, access to Linked Data still requires substantial technical skills, thus denying use of these information resources to users with no technical know-how. This thesis argues that usable tools can be built for non-technical users, ones without any programming skills, that use the affordances of Linked Data to meet information needs that require combining and querying data from several information sources. The thesis accomplishes this goal by taking a user-centred approach to derive requirements for building such tools, and developing a series of interactive prototypes, each of which builds on the results of the previous attempt that advances the usability of end-user data-centric tools. By developing tools that use structured data published on the Web rather than unstructured sources, this thesis provides a substantial improvement over the current breed of tools used to support combining and querying distributed information sources, which often require an substantial time and effort to accomplish this kind of tasks.

1.1 Problem Definition and Scope

Structured data sources are increasingly available on the Web; however, as we pointed out, some tasks require data to be combined from several sources in order to solve a data-centric information need. Relative to the whole set of information-seeking activities

⁴<http://linkeddata.org>

performed on the Web, these kind of data-centric questions belong to the *long tail* of information needs - the property that rare information needs cumulatively rivals the number of occurrences of commonly occurring information needs. Thus data-centric needs are so specific that they rarely warrant creation of a dedicated Web page or an application. Therefore this kind of queries can only be supported by enabling real-time data-centric interactions. Linked Data provides a technical framework that aims at making data integration on demand easy, by a priori providing links between data sources as part of the publishing process. However, deploying such tools in the real world, includes both technical and interaction challenges.

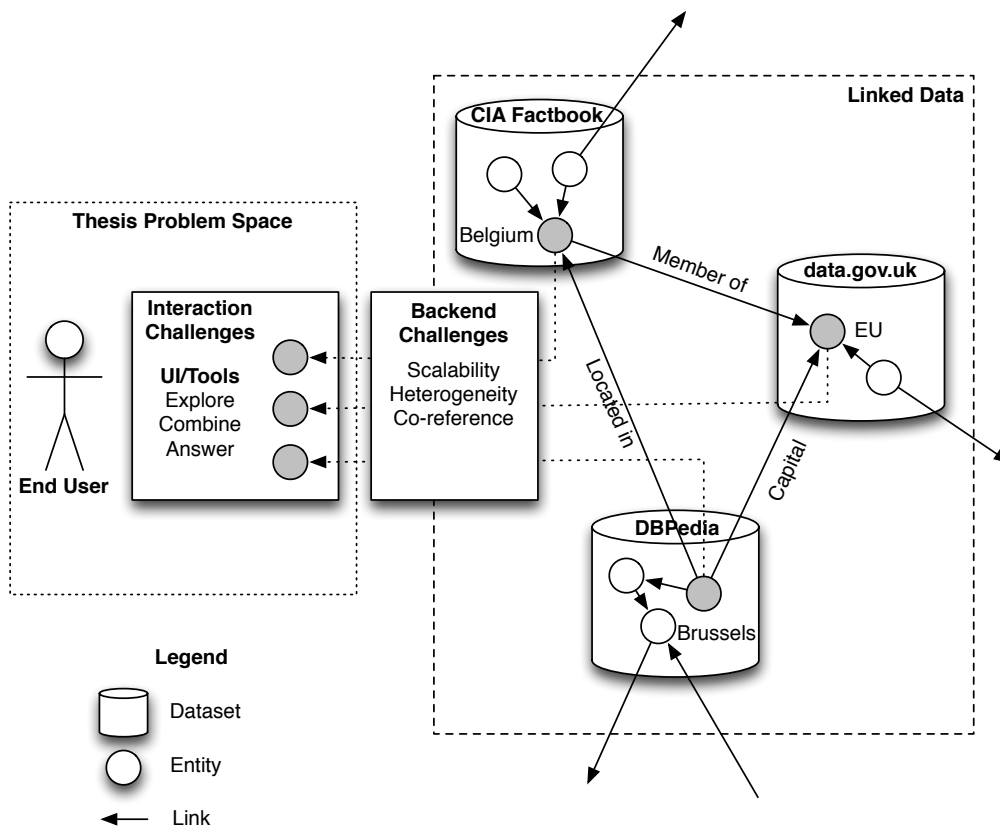


Figure 1.2: Scope of thesis from the whole set of challenges of delivering data browsers over Linked Data.

The technical challenges of enabling end-user tools over Linked Data include issues of scalability and data heterogeneity. The fact that Linked Data is envisioned as a distributed system results in scalability issues with respect to querying data from distributed datasets. Since real-time interaction requires quick and responsive query times scalability remains a crucial requirement. Research into *federated queries* - queries that run over several datasets - attempt to provide solutions to this problem (Schwarte et al. (2011); Ladwig and Tran (2010)). Data heterogeneity, on the other hand, can introduce data consistency problems in the interface (Euzenat and Shvaiko (2007)). Terminological heterogeneity or co-reference problems relate to identifying if two entities are the

same which can largely depend on the context; for example if the state of Germany as an entity is the same with an entity describing Germany during World War II. Structural heterogeneity, on the other hand, refers to having different schemas for the same data; for example having a single string for an address in one dataset versus separate fields for road, address number and post code in another. Supporting users who deal with issues of heterogeneity can be useful for better supporting users with some skills in data representation; however more often than not it is far too complex for average end users.

This thesis is concerned with the interaction challenges of providing tools that enable combining Linked Data to answer information needs. The scope of this thesis is depicted in Figure 1.4. As the Figure shows, even if the technical challenges associated with Linked Data are largely solved, how end users, those without technical knowledge, can access the data still remains a challenge. By users without technical knowledge we denote users without programming or database skills; this kind of users have knowledge of interacting with data only to the extent of using end user data tools such as spreadsheet applications. This thesis looks at one specific case of tools for end users: tools that allow data to be combined and queried in real time to solve data-centric needs. It attempts at identifying the requirements for enabling such tools, and suggests solutions to interaction challenges by developing several interactive prototypes that address these challenges.

1.2 Research Challenges

The problem of developing usable tools for end-users that leverage Linked Data for exploring, finding, and combining data to solve data-centric needs can be broken down into the following research challenges:

RC1. What are the requirements for developing usable end-user data-centric interactions over Linked Data?

As with any novel design space, a key challenge is identifying what are the core requirements or the core problems that need to be solved in order to have usable interfaces that allow data-centric interactions over Linked Data. Eliciting a list of requirements also serves as a benchmark for comparing various end users Linked Data tools. Additionally it allows us to examine how existing solutions on data-centric interactions apply over the problems associated with tools using Linked Data. For example, how does research related to data-centric tools coming from the Human-Computer Interaction and Information Retrieval fields inform the design requirements of data-centric tools over Linked Data? How does the current breed of mashup tools differ from mashing Linked Data sources? What are the differences related to research on interfaces from the database community? Finally, can these requirements inform on additions to Linked Data as a technology in its present form and conversely how does Linked Data as a particular

technology and method of publishing data currently influences the design of data-centric tools.

RC2. How do we represent graph-based structured data to non-technical users?

An immediate problem in providing end-users access to Linked Data is how data is represented to users. The choice of data representation is influenced by several factors. First, data representation is influenced by how generic the tool is by design, i.e. if the tool allows immediate data representation without the need of additional formal descriptions (e.g. templates). This allows Linked Data to be accessed with the tool on demand, without any additional configuration, relying only on human-readable values in the data for rendering a representation in the tool.

Data browsers such as Disco⁵, for example, displays one entity per page, which includes information about each property and property value to which the entity links. The page also includes information about other entities that has link to the entity being inspected. Early browsers focused on publishing RDF, such as IsaViz⁶, directly portrayed the underlying structure of the data being accessed by providing graph visualisations in the interface. Browsers such as Tabulator, on the other hand, visualise graphs by displaying entities in nested tables (Berners-lee et al. (2006)). For facilitating richer representations of Linked Data, other tools introduce the concept of lenses - a template-like descriptions that inform the interface how to render the data. Lenses vary based on the granularity of data they render - some are used to just render entities in the dataset, while others can take entire subsets of data and be used as a widget-like representation.

If the tool provides exploration capabilities i.e. are able to pull in related data sources on demand, a suitable representation is needed to represent this process. The Tabulator for example uses a nested table metaphor to display navigation from one entity to another (Berners-lee et al. (2006)). Parallax, a browser over Freebase⁷ introduces set-oriented browsing, where each navigation step in a graph is done simultaneously with multiple entities that share a common property (Huynh and Karger (2009b)). This form of navigation is demonstrated by representing successive sets of entities as a collection in a new page. While this type of "data navigation" is a core concept of data browsers, there have been very few studies evaluating various approaches to representing data navigation.

Finally, data representation is also informed by the purpose of the tool. Tabulator and Parallax, for example, are designed to empower the user to engage in sense-making activities over Linked Data. Both tools allow explored data to be presented in various representations such as maps, timelines, charts etc. Other tools may have been built with

⁵<http://www4.wiwiiss.fu-berlin.de/bizer/ng4j/disco/>

⁶<http://www.w3.org/2001/11/IsaViz/>

⁷<http://freebase.com>

other motivations in mind. For example, tools such as gFacet or BrowseRDF provide end-users with a data querying interface over Linked Data - thus both interfaces build upon the familiar hierarchical faceted browsing paradigm to provide queries over Linked Data (Heim et al. (2010); Oren et al. (2006)).

RC3. How do we let users quickly interrogate and find the data they need in unfamiliar datasets?

Linked Data is often described as one global distributed database; however for the most part it is structured around datasets - Linked Data repositories, published and maintained by a single publisher. When searching for data one might find a relevant dataset that can potentially contain the data needed to answer a data-centric need, however, often the case is that these datasets themselves still contain data on a vast number of topics. Thus, similar to consuming information from Web sites, often users will only need a small fraction of the data from the overall dataset. This means that mechanisms are needed that facilitate efficient interrogation of Linked Datasets.

This problem can be broken down in several parts. The first deals with how users begin exploring a dataset. For example, in Tabulator the search begins by inputting a machine readable URI (Berners-lee et al. (2006)). Clearly, users have little knowledge of a machine-readable URI representing a description of a real world object, and even less any ways to find them. Other tools, such as Parallax and gFacet begin with list of the ontology or schema level concepts of the dataset, displaying these as collections of entities in the interface (e.g. People, Presidents, Cars etc.) (Huynh and Karger (2009b); Heim et al. (2010)). Some tools, such as VisiNav rely on keyword search to surface both ontology concepts and specific entities (Harth (2009)).

More important than the exploration starting point is how users explore a dataset to find the subset of data needed to answer an information need. The solution needs to provide users with tools to query a dataset. While various approaches exist to querying structured data, this thesis is concerned with direct manipulation approaches - approaches that empower the user to directly query a dataset through manipulating objects in the interface (Shneiderman (1983)). Faceted browsers are an example of direct manipulation tools for exploring single-concept collections of information (Yee et al. (2003)). Data browsers are direct manipulation interfaces designed to query multi-concept collections of information such as graphs of data. Direct manipulation interfaces are also better suited for exploratory queries.

Direct manipulation tools for querying Linked Data vary substantially in design. For example, the gFacet interface represents navigation of a graph dataset as building a custom hierarchical faceted browser (Heim et al. (2010)). Parallax uses *set-oriented navigation*, a technique of navigating multiple entities simultaneously through a common property, to facilitate similar exploration (Huynh and Karger (2009b)). Tabulator uses a query-by-example approach - users can expand (or navigate) one entity at a time;

the interface then allows patterns to be specified along this trail of navigation to find similar data (Berners-lee et al. (2006)). The problem with navigational direct manipulation interfaces is that the scale of graph datasets is sometimes so large that relying on navigation to find how entities are connected can be viewed as navigating a maze with no map.

RC4. How do we deal with information overflow?

Even with a good exploration tool that abstracts machine-readable data and allows users to perform various queries, datasets can still be difficult to explore. They can hold enormous amounts of data, thus frequently requiring users to find and filter through a small portion of the dataset. This problem is particularly challenging when users are engaged in an exploratory search - search where users have no specific information goal but rather engage in exploration. For example, users might want to see what interesting data they can find about the major cities in the UK. Thus users are left with the task of figuring out which properties would make sense to combine, visualise, or which properties would make good facets for filtering. Doing such iterations can be very time-consuming in a generic tool. The authors of BrowseRDF were the first to take note of this issue by trying out an automatic way of detecting useful facets to combine (Oren et al. (2006)). They acknowledge, however, that automatic approaches are limited and that additional knowledge about the ontology is required. While on the long run data-centric browsers could analyse graphs of data and offer recommended views based on established ontologies, such capabilities are not feasible in the foreseeable future.

RC5. Can we leverage the usability of custom made applications designed over fixed datasets to facilitate data-centric interactions that are typically provided by less usable, generic data-browsing applications?

If a data-centric tool is more generic by design, it allows immediate access to Linked Data sources; however it does so by usually deploying a generic representation scheme for displaying the data e.g. tables or simple lists. Thus while data can be accessed through the tool quickly and on demand, generic data tools tend to be less appealing or more "geeky" than custom-made applications which are created and customised around a fixed and known sources of data. For example, for finding nearby restaurants at a selected location a custom-built application or a search on Yelp⁸ could provide that information by simply entering a location. If raw data is available, the same query could be answered by using a generic data browser, although at a much higher cost. On the other hand if you want to view the locations of restaurants in relation to bus stops, a custom application not providing bus information would be unable to answer that question even if raw data is available about bus stops. At best we would need to use another application and do the data integration manually. If Linked Data was available,

⁸<http://www.yelp.com>

however, a generic browser could be hypothetically used to merge the data on demand and find a suitable answer.

The problem of both approaches is that they sit at the end of two extremes - one is very versatile yet not as usable, while the other is very usable yet not versatile. Thus the challenge is how to find a solution, one that retains the usability of custom applications, and benefits from the interoperability of merging sources of Linked Data. Several approaches have been proposed. Tim Berners-Lee Design Notes on a Semantic Web Clipboard⁹ envisions applications able to share data with each other. The approach described, however, requires a substantial amount of effort including writing data converters, accounting for different ontologies and data vocabularies and rules. Another possible approach is to use configurable data-mapping approaches. These allow applications to be configured and set up over data without any programming. This approach, however, is often inapplicable for end users since it often requires technical know-how and the configuration process is too time-consuming to be able to do so on demand.

1.3 Approach

The approach this thesis takes in solving the problem of providing data-centric interactions over Linked Data is divided into three areas:

- Deriving requirements for end-user data-centric tools. Identifying the areas that present interaction challenges for end-user use of Linked Data for data-centric tasks;
- Supporting data-centric interactions for non-technical data users - users with certain knowledge in end user data tools such as spreadsheets, but no knowledge in database technologies or programming. Such users can be supported with generic data tools such as data browsers;
- Supporting data-centric interactions for casual Web users - users that only experience data through rich Web applications. Such users can be supported by closely associating data-centric interactions with how they use the Web today.

1.3.1 Deriving Requirements for End-user Data-centric Tools

This thesis first approaches the problem of delivering usable data-centric tools over Linked Data by using scenario-based design and presents examples of personas in order to derive high-level requirements for data-centric interfaces. Personas are hypothetical descriptions of specific potential users along with scenarios of typical problems they

⁹<http://www.w3.org/DesignIssues/SemanticClipboard>

encounter and current solutions that are used to solve these problems. The scenarios described through personas can be then broken down into specific requirements that a system needs to provide in order to facilitate data-centric interactions.

This thesis identifies four core requirements for delivering data-centric tools over Linked Data:

- **Discoverability.** Provide a suitable way to find data sources from publishers over the Web as a starting point of exploration.
- **Legibility.** Represent data that facilitates understanding about what data is available, and how this data is linked throughout the graph.
- **Triage.** Allow versatile interactions which enable efficient triage/exploration/browsing of data sources.
- **Tools for Analysis and Sense-making.** Supply the user with easy-to-use tools of displaying data in various ways that facilitate sense-making.

The thesis then puts forward an analysis on how different data-centric tools approach these core requirements. At the same time it examines relevant work coming from the Human-Computer Interaction community and discusses potential applicable solutions to data-centric problems, as well as solutions that tend to be specific to Linked Data and thus need different solutions.

To explore these challenges in further detail, this thesis first focuses on supporting a particular group of users - spreadsheet users - end-users knowledgeable in manipulating spreadsheet data, but have no knowledge in advanced technical skills such as database querying or programming. Spreadsheets have gained wide popularity among non-programmers both because of their ability to perform custom computations with data as well as support sense-making tasks (Nardi and Miller (1990); Russell et al. (2008)). As S. Hudson notes (Hudson (1994)):

“Spreadsheets are one of the few true success stories among systems for end-user programming - that is, systems designed to allow non-programming users to create computations of their own design”.

Recent use of tabular representations of data has also been picked up by online communities outside the professional arena. For example, social data sharing sites such as ManyEyes¹⁰ allow users to input tabular data, visualise it in numerous ways and discuss the results with peers (Viegas et al. (2007)).

¹⁰<http://manyeyes.alphaworks.ibm.com/manyeyes/>

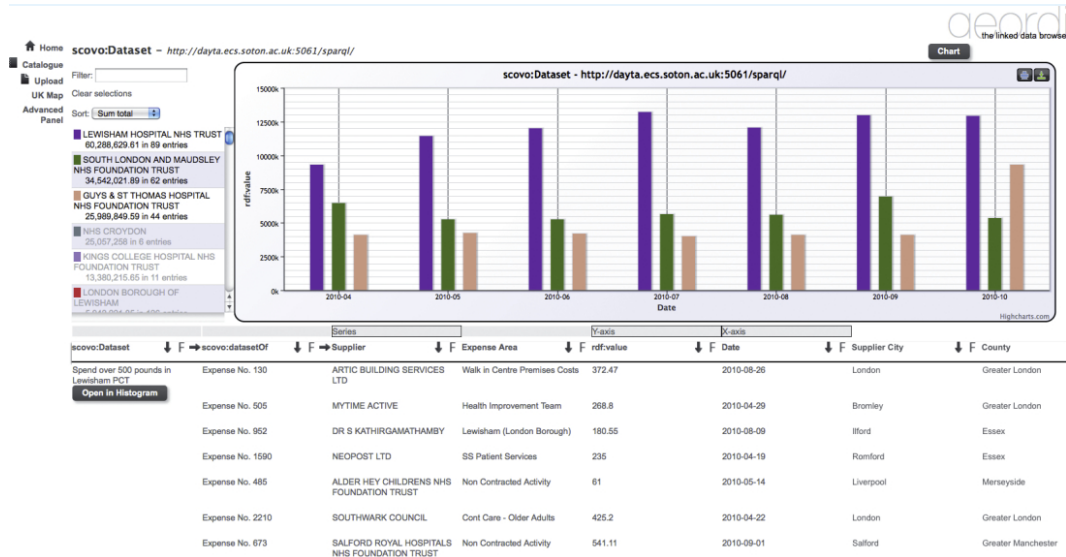


Figure 1.3: The GEORDi generic data browsers showing a spreadsheet generated from Linked Data and a graph widget to visualise the data.

In order to explore the challenges of bridging the gap between Linked Data and spreadsheet use, this thesis goes on to present a generic-data browser named GEORDi (Figure 1.3) which builds upon a spreadsheet metaphor to support both data exploration and representation. In order to explore graph datasets, GEORDi uses an unfolding column metaphor - each column constitutes a set of entities, and a new column could be added by selecting a property which is shared by the entities in the column. Because graphs are not tabular in nature, a tree-based, nested tabular representation is used in order to show the relationships between entities described in different columns. Once users generate a spreadsheet of data they need, a number of visualisation tools, such as charts and maps, are available to make sense of the data.

1.3.2 Visor: Exploring Unknown Data Domains

The deployment and testing of GEORDi over several publicly available Linked Datasets unveiled a number of challenges with data browsers in general as well as with some of the approaches used in GEORDi.

First, the tree-nested spreadsheet visualisation proved difficult to comprehend beyond several columns. This was particularly the case when the tool was used to explore and find data in an unfamiliar dataset. In this case, the number of columns grew increasingly large and it was difficult to follow which column was derived from which previous column. Second, as with other browsers, the concept of browsing graph data did not prove useful for exploring unknown graph datasets, i.e. when the number of concepts in the ontology of the dataset is large. For example, it is difficult to solely rely on browsing when one needs to find different ways that two entities or sets of entities are connected through

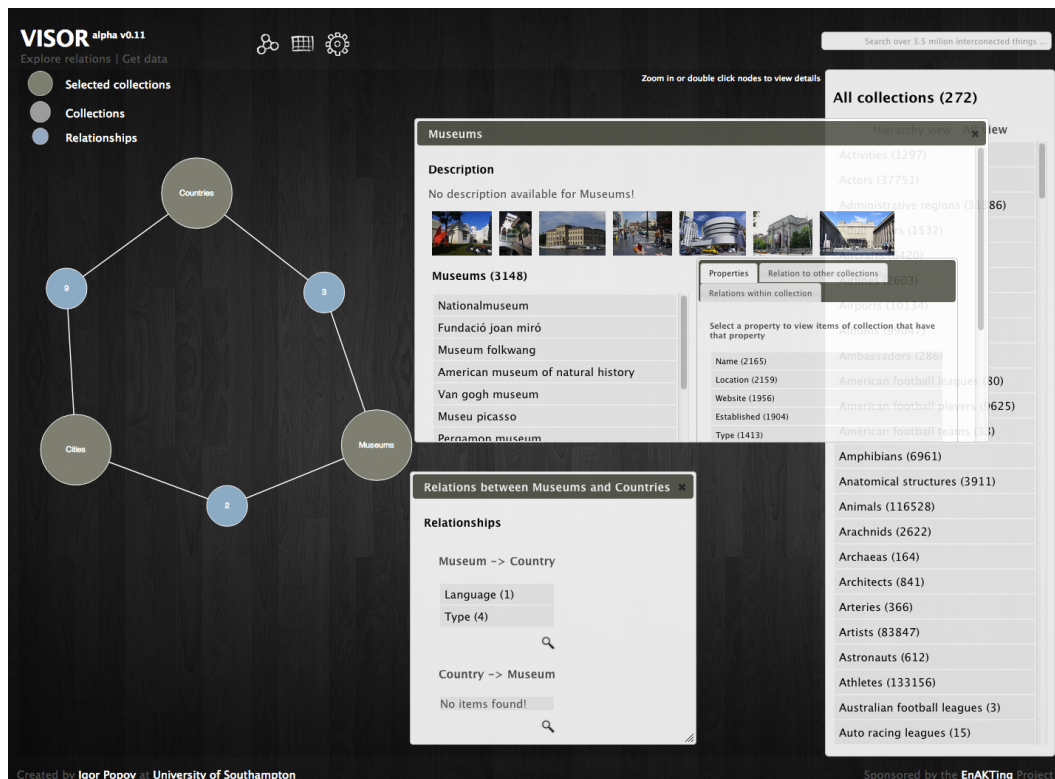


Figure 1.4: The Visor browser offering an overview-first, instance data on demand way of browsing a Linked Dataset.

the graph. Such queries are common if a user is unfamiliar with the content or schema of a dataset. Thus it became apparent that users would need additional tools that go beyond navigation to support exploration of unfamiliar datasets.

To tackle these challenges this thesis presents a tool called Visor, which extends navigational exploration with multi-pivoting - a technique to help the user explore unfamiliar datasets. Visor includes several concepts that sets it apart from traditional end-user Linked Data tools. First, exploration can be initiated by selecting multiple items of interest i.e. a user can select different sets of entities through selecting concepts from the ontology. The system then helps the users to semi-automatically find how selected concepts can be linked. Second, the interface follows “an overview first, instance data on demand approach. This allows users to quickly gain an understanding of the different types of data in a dataset and how these are linked without being overloaded by instance data. Additionally, unlike other data browsers that allow only forward navigation, Visor allows links to be used in both directions. Once the required data is identified the user can specify which parts of the data should be used to create a spreadsheet which can be exported.

1.3.3 mashpoint: Linking Data-centric Web Applications

Both Visor and GEORDi are built on the assumption that a large number of users familiar with end-user data technologies, such as spreadsheets, can be supported with tools enabling data-centric interactions over Linked Data. Still, a majority of the users on the Web today are not spreadsheet users - these are users that interact with data through visually rich and custom-built Web applications. In order to support data-centric interactions for casual Web users this thesis presents a mashpoint, a framework that allows distributed data-centric Web applications to be linked based on the similarities of the entities in their datasets. Once integrated into the framework, it allows users to pivot with an arbitrary set of selected entities from one application to another - essentially enabling data-centric navigation on the Web. This navigation is simply presented to the user as an extended version of Web navigation between Web documents. By using data-centric navigation, users can easily perform many of the interactions needed for querying information from distributed data sources, without having to manually extract data from the Web pages or use a generic data browser.

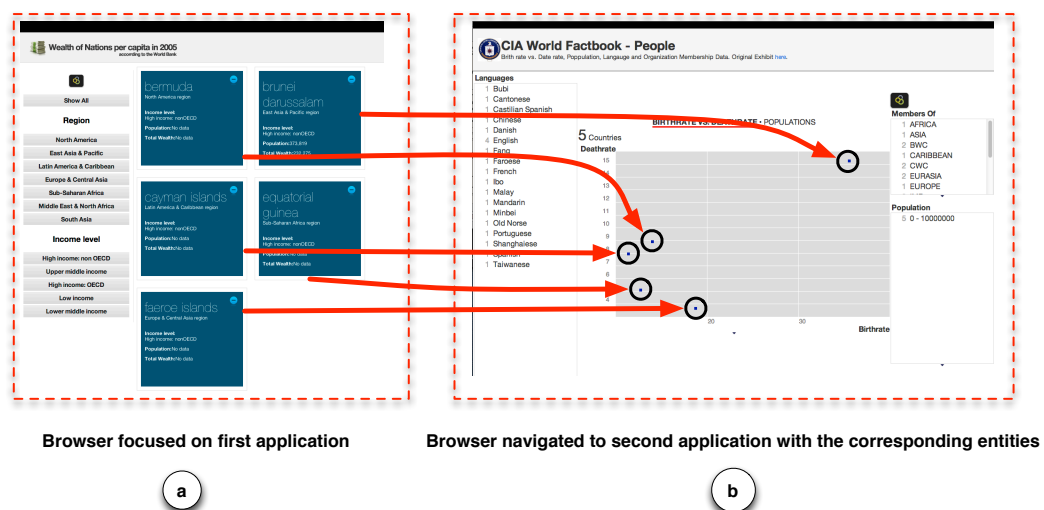


Figure 1.5: Browsing for one data-centric Web page to another with a set of entities.

Figure 1.5 shows the concept of data-centric interaction between two data-centric applications on the Web. The first application (1.5a) displays GDP per capita data about countries. The Figure shows that the set of all the countries is filtered to a set of five countries. Data-centric navigation in mashpoint allows the selection of entities made in one application to be used to refocus on a particular set of entities in another data-centric application (1.5b), or in this case CIA factbook data about countries.

mashpoint is built on the idea that there are many data-centric applications on the Web that inherently talk about the same entities in their datasets. If an application displays certain data about entities and then asks for other data about the same entities

in another application, it would be essentially equivalent to navigating raw Linked Data. In this case, however, the user now has the benefit of viewing the data through a lens of a custom made application. From this perspective, applications on the Web of Data can be viewed as lenses offering higher level abstractions over a certain subset of Linked Data. By allowing these lenses to communicate the entities they describe in the application, mashpoint can support data-centric operations for end users without any need of using a generic browser.

1.3.4 Scope and Limitations

This thesis is concerned with the interaction challenges associated with accessing, exploring and manipulating vast amounts of graph data. As we discussed, many back-end challenges persist in delivering fully functional data browsers that can operate over a truly distributed Web of Data. Thus, while the solutions in this thesis are currently capable of operating over single large graphs of data they are designed with the intention to support browsing of truly distributed and heterogeneous data on the Web when the corresponding back-end infrastructure technology matures. In particular this thesis makes the following two assumptions:

- In this thesis I assume that problem of data heterogeneity is one that requires solutions at the back-end before generic browsers can truly operate over distributed datasets. While user interfaces that expose heterogeneity and support non-programmers in data reconciliation tasks do exist, this task can be supported in a very limited fashion and is usually beyond the scope of a truly casual user. In order to deliver truly usable data browsers, reconciliation needs to be supported and resolved before data is accessed by a browser.
- Most of the experiments and prototypes developed in this thesis either operate over datasets that can be queried or can be accessed in a fast and reasonable time while providing sufficient fidelity of accessing large amounts of graph data. In order to truly access the Web of Data, however, a web browser will need to query multiple distributed data repositories simultaneously. These types of federated queries, however, are currently relatively slow for the kind of responsiveness we require in a data browser.

1.4 Contributions

The statement of this thesis is:

Usable tools can be built over Linked Data on the Web that allow non-technical end users to answer information needs that require combining and querying data from several information sources.

The thesis makes the following contributions:

- First, it provides a map of the core interaction challenges associated with using generic data-browsers over Linked Data.
- Second, it contributes multi-pivoting, an approach to improve exploring unfamiliar linked datasets, by assisting users in finding relationships between arbitrary selections of data within a dataset. With multi-pivoting we offer a way of users to quickly find and focus on the specific data types and relationships within a dataset which are relevant to answering a data-oriented query.
- Third, it contributes mashpoint, an approach to linking existing applications built over Linked Data that allows data-centric navigation between applications. By allowing data-centric navigation between web applications, the approach offers a solution to the challenges of rich data representation and information overload, associated with generic data-browsers.
- Fourth, by offering a lightweight approach to linking applications, this thesis demonstrates that rich interactions can be provided to end users without the need of data publishers to implement the full complex stack of Semantic Web technologies.
- Finally, by developing and studying various systems that provide tools, which allow users to explore and answer questions by combining Linked Data, this thesis provides a map of interaction challenges and best practices associated with building data-exploration interfaces over Linked Data for non-technical users.

1.4.1 Publications relating to thesis work

The following publications were published as a result of the work presented in this thesis:

Put in your postcode, out comes the data: A Case Study

Tope Omitola, Christos L. Koumenides, Igor O. Popov, Yang Yang, Manuel Salvador, Martin Szomszor, Tim Berners-Lee, Nicholas Gibbins, Wendy Hall, m. c. schraefel, and Nigel Shadbolt. *Put in your postcode, out comes the data: A case study*. In ESWC (1), pages 318-332, 2010.

Note: This paper won best paper award at the Semantic Web In-use Track at ESWC 2010.

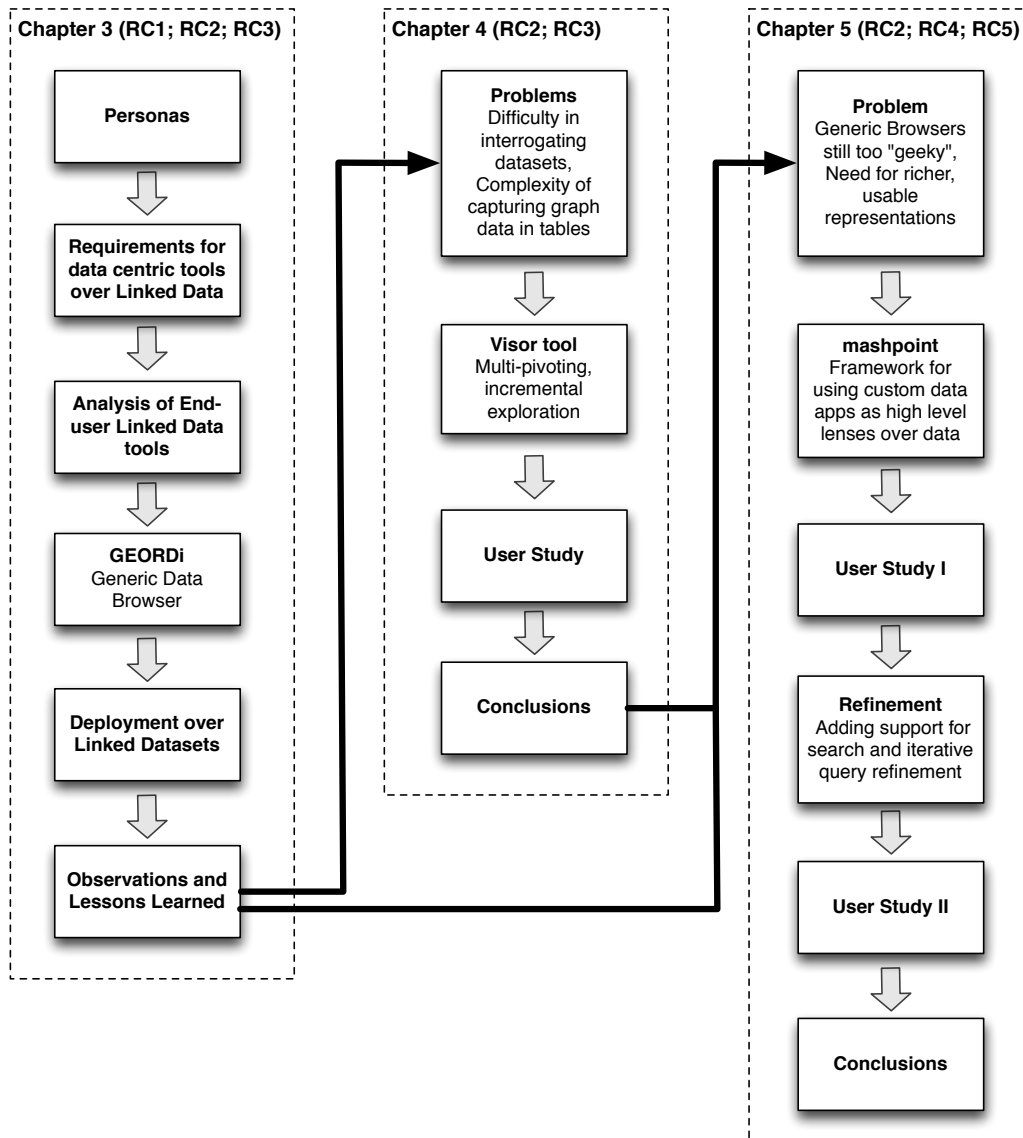


Figure 1.6: Thesis chapter organisation based on work conducted.

Accessible at: <http://eprints.ecs.soton.ac.uk/18765/>

Data Picking Linked Data: Enabling Users to create Faceted Browsers

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

Accessible at: <http://eprints.ecs.soton.ac.uk/20804/>

Will this work for Susan? Challenges for Delivering Usable and Useful Generic Linked Data Browsers

schraefel, m., Smith, D., Popov, I. , Van Kleek, M. and Shadbolt, N. (2010) *Will this work for Susan? Challenges for Delivering Usable and Useful Generic Linked Data*

Browsers. Technical Report , School of Electronics and Computer Science, University of Southampton.

Accessable at: <http://eprints.ecs.soton.ac.uk/21967/>

Connecting the Dots: A Multi-pivot Approach to Data Exploration

Popov, I., schraefel, m., Hall, W. and Shadbolt, N. (2011) *Connecting the Dots: A Multi-pivot Approach to Data Exploration*. In: International Semantic Web Conference, 23-27 October 2011, Bonn, Germany. (In Press)

Accessable at: <http://eprints.ecs.soton.ac.uk/22784/>

GEORDi: Supporting lightweight end-user authoring and exploration of Linked Data

Popov, I., Smith, D. A., Van Kleek, M., schraefel, m., Correndo, G. and Shadbolt, N. (2010) *GEORDi: Supporting lightweight end-user authoring and exploration of Linked Data*. Technical Report , School of Electronics and Computer Science, University of Southampton.

Interacting with the Web of Data through a Web of inter-connected lenses.

Popov, I., schraefel, m., Correndo, G. Hall, W. and Shadbolt, N. *Interacting with the Web of Data through a Web of inter-connected lenses..* In, WWW2012 Workshop: Linked Data on the Web (LDOW2012), Lyon, FR, 16 Apr 2012. 9pp.

Accessable at: <http://eprints.soton.ac.uk/336573/>

marshpoint: Supporting Data-centric Navigation on the Web.

Popov, I. *marshpoint: Supporting Data-centric Navigation on the Web*. Poster Presentation, At CHI 2012, Austin, Texas, 05 - 12 May 2012..

Accessable at: <http://eprints.soton.ac.uk/273237/>

marshpoint: Browsing the Web Along Structured Lines.

Popov, I., schraefel, m., Hall, W. and Shadbolt, N. (2012) *marshpoint: Browsing the Web Along Structured Lines*. At UIST (ACM Symposium on User Interface Software and Technology) 2012, Cambridge, US, 07 - 12 Oct 2012.

Accessable at: <http://eprints.soton.ac.uk/342523/>

1.4.2 Other Publications

Trust Me, I'm Partially Right: Incremental Visualization Lets Analysts Explore Large Datasets Faster

Fisher, D., Popov, I., Drucker S., and schraefel, m. (2012) *Trust Me, I'm Partially Right: Incremental Visualization Lets Analysts Explore Large Datasets Faster*. In: CHI 2012

Accessible at: <http://eprints.soton.ac.uk/273149/>

1.5 Thesis Overview

The rest of the mini-thesis is structured in six chapters:

Chapter Two: Background and Related Work Following the introduction, Chapter 2 discusses background topics and related work in the space of end-user interfaces over Linked Data. First, it introduces the concept of structured data and the use of structured data to power new methods of information exploration. Second, it provides an overview of the evolution of structured data on the Web, starting from database-powered Web sites and embedded data on Web pages to the vision of the Semantic Web and a Web of Data. Finally, it maps the broader area of end user tools over Linked Data, including tools for data exploration, querying and data visualisation.

Chapter Three, Four and Five present solutions to the research challenges stated in the Introduction. The structure of each Chapter is shown in Figure 1.6.

Chapter Three: Design Process for Generic Linked Data Browsers Chapter Three starts off by suggesting different personas as a way to elicit core requirement for data-tools that allow data interactions over Linked Data. Using these requirements as a benchmark, it examines existing end user Linked Data tools and interaction techniques particularly focusing on generic data browsers. It then describes GEORDi - a generic browser aimed at users familiar with spreadsheets. GEORDi associates all aspects of data-browsing using a spreadsheet metaphor - it uses a tree-based nested spreadsheet visualisation to represent graph data as well as uses unfolding columns as a way to navigate and discover new data. The Chapter describes the lessons learned from deploying GEORDi over a number of Linked Datasets.

Chapter Four: Multi-pivot Exploration of Data on the Web Chapter Four presents Visor, a data-browser that implements *multi-pivoting*, a method aimed at improving the shortcomings of data navigation as a mechanism for exploring unfamiliar Linked Datasets. Visor also uses “an overview first, instance data on demand approach, allows bi-directional navigation and allows generation of custom spreadsheets. The Chapter provides the motivations for this approach, describes the user experience and provides implementation details. Finally, it presents the evaluation results from an exploratory user study that was conducted with Visor to ascertain the viability of the approach.

Chapter Five: mashpoint - Browsing the Web along Structured Lines Chapter Five tackles the problem of supporting data-centric interactions for casual end users - those that experience data only through the lens of a rich Web application and have no

familiarity with spreadsheets. I present mashpoint - a framework aimed at linking data-centric Web pages based on the similarities of their entities, which in turn allows Web browsing to be extended with data-centric behaviours. It also describes the lightweight URI-only approach of the mashpoint architecture. In addition to supporting casual end users the Chapter uses mashpoint lightweight approach as an economic argument about what can be achieved by using only the minimal amount of Semantic Web technologies.

Chapter Six: Discussion: Implications for Design Chapter Six reflects on the tools successes and future directions for these tools as the field of Linked Data evolves and matures.

Chapter Seven: Conclusion and Future Work Chapter Seven reviews the contributions of this thesis and articulates a vision for the future of data-centric interactions on the Web.

Chapter 2

Background and Related Work

This thesis investigates how to design data-browsing interfaces that allow end users, those without technical knowledge in databases, formal query languages or programming, to use Linked Data to solve data-centric information needs that are otherwise difficult to solve using unstructured, text-based documents on the Web. This Chapter serves to provide a background to a number of relevant topics. First, it lays out a definition of structured data and discusses the affordances of structured data in providing better access to information. Second, it describes the current sources and uses of structured data on the Web. This is followed by a discussion of the notion of a Semantic Web and a Web of Linked Data, as a proposed evolution of publishing structured data on the Web.

The Chapter then turns to the implications of having a Web of Linked Data on end-user data-centric interactions. First, it provides use cases for end-user data-centric interfaces over Linked Data and briefly discusses the core challenges of building such interfaces. This is followed by an overview of a number of active research areas centred on data interfaces for end users. The Chapter then provides an overview of existing designs of data-centric interfaces over Linked Data. The Chapter concludes with a discussion on limitations and challenges of reviewed approaches.

2.1 Structured Data

In a world of exceedingly accessible information, users rely on evermore sophisticated search interfaces and algorithms to help us find and filter out needed information. While keyword search predominates in finding documents with relevant content, we often lend ourselves to more specialised form of search, particularly when we wish to query for more granular sources of information. Take, for example, the task of buying a phone on a typical online shopping site, as shown in Figure 2.1. Commonly our first task would be to try and narrow down the vast information about various phones based on

certain preferences, such as price, manufacturer, and carrier. The ability of systems that allow us to filter down information according to different facets as shown in Figure 2.1, is due to the fact that the information about phones is structured in a certain format. The term structured data refers to information represented in a way that there is identifiable format in which the data is described. In most cases, besides the information itself, additional information (also known as *metadata*) is provided. In this example, information about phones is somewhere explicitly stated - this allows a machine to “know” which part refers to the price and which to the manufacturer. Data can be structured in various formats. Spreadsheets are another common example of structuring information, where each column name represents the kind of information contained in each cell of the column. Relational databases structure and store information in multiple tables with relationships defined between rows in each table. The Web today is increasingly becoming just a lens through which we interact with structured data. A majority of Web sites are powered by backend databases over which web pages and applications of content are presented to users. These databases that power web sites are sometimes referred as the Deep Web (He et al. (2007)). While estimates vary, most conclude that the structured information contained in the Deep Web today is several orders of magnitude larger than the surface Web.

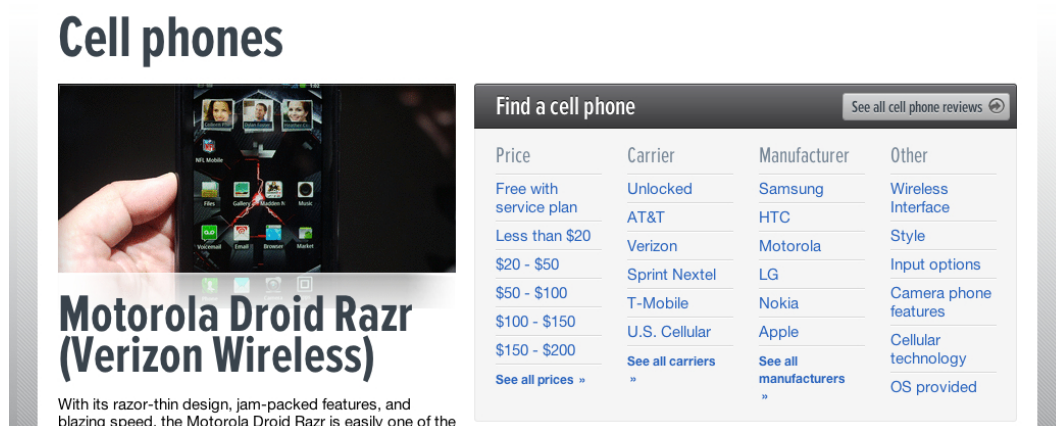


Figure 2.1: A typical online shopping site offering data-centric features to filter for information.

2.1.1 Uses of Structured Data in Information Exploration

Structured information can be deployed in a variety of formats to serve different purposes. For example, collections of text documents or Web sites can be assigned metadata in order to convey the general content of the document. Another example would be a web site organising its site hierarchy to improve navigation to relevant documents on the site. Online digital libraries, for example, provide metadata on top of their documents to enable easy access and better document retrieval. Apart from adding structured information to documents, the biggest source of structured data comes from databases

that structure content around real-world objects on a much finer granularity. A large portion of Web applications today (e.g. online shopping sites, social networking sites etc.) use some sort of database, which is queried through structured languages, over which a layer of application logic controlled from a website provides a bridge to access the data.

Deploying information in structured format allows new ways for end-users to interact with information. Structured content is most commonly used to support advance search and query features that differ from keyword search queries. Both types of search have their strengths and weaknesses. Keyword search allows freedom of expression and a relatively simple interaction model - the user enters keywords and is given a list of relevant results. Structured search allows content to be filtered according to specific criteria. Both keyword and structured search have been extensively used in different areas; while keyword search has predominately been used to search the Web, the latter has been employed for browsing more contained collections of data on data-powered websites (e.g. online shopping sites). Additional affordances of structure in information include the ability of retrieved information to be more easily visualised and represented in multiple ways. Other uses include the ability to perform data-centric operations, such as sorting. In the following section we discuss the areas where structured information is used to support information interactions.

2.1.1.1 Organisation of Information Content

One of the most common uses of structured information is for organising digital content; for example, organising information in categories and hierarchies. A large number of Web sites, for example, provide content on a variety of subjects and topics. A common feature on such Web pages is to provide some categorisation of the content that allows users to quickly narrow down information to the particular content they are interested in. For example, a site offering media content might provide categorisations such as music, movies, TV shows etc. The simplest type of categorisations is flat categories where every item falls into exactly one category. Hierarchical categories, on the other hand, allow for association with multiple categories, where the categories are arranged in a hierarchy. In a hierarchical ordering of information each item of information is contained in one path in the hierarchy. Hierarchies are still used today to navigate large online information content providers. In the early days of the Web, searching for web pages that have previously been categorised in hierarchies was also commonly used. Directories of web sites such as Yahoo¹ and later on the Open Directory Project (ODP)² were popular ways to find content on the Web (Pollock and Hockley (1997)). As the growth in the number of Web sites increased and with the emergence of reliable search engines the importance of general web directories declined.

¹<http://www.yahoo.com/>

²<http://www.dmoz.org/>

2.1.1.2 Faceted Browsing and Filtering

Often documents can discuss multiple things simultaneously. In such cases, documents do not naturally fit in any single hierarchy. Faceted classification allows documents to be assigned values of any category of a predefined set of categories (or *facets*). For example, when given a collection about cars, each car can be assigned one or multiple values of a list of categories; in our case these might be the manufacturer (BMW, Mercedes, Ford, etc.), colour (red, white, grey etc.), and price (10000£- 20000£, 20000£-30000£, etc.). [Hearst et al. \(2002\)](#) describe faceted metadata as being composed of “orthogonal” sets of categories. Each category describes an independent aspect of the information item. Faceted categorisation is often attributed to for introducing the idea with his colon classification system, which suggested describing information items by multiple classes ([Ranganathan \(1933\)](#)). [Bates \(1988\)](#) supported faceted library catalogue representations in the 1980’s. The use of a faceted classification for browsing, however, was first pioneered by [Allen \(1995\)](#) for browsing document collections that did not naturally fit together to form a single hierarchy.

A faceted browser allows narrowing down the collection of items by selecting values associated with a facet. Using a faceted browser, a selection of a value in a certain category filters for documents that have the relevant value assigned for that particular facet. When a particular value of a facet is selected, the rest of the facet choices get updated to reflect the remaining choices for narrowing down the current list of items. Every subsequent selection filters down the resulting set of items to according to the selected values. In essence, faceted browsing allows users to custom build their own hierarchy on the fly.

Faceted browsing has been extremely popular on the Web and is commonly adopted in many online shopping sites because different users have different criteria for choosing products. Figure 2.2, for example, shows the Flamenco faceted browser for browsing collections of art works ([Yee et al. \(2003\)](#)). Research studies around faceted navigation have shown both improved task performance and user preferences of using faceted browsers instead of keyword search when browsing information collections ([Hearst \(2000\)](#); [Hearst et al. \(2002\)](#); [Yee et al. \(2003\)](#); [Hearst \(2006\)](#)).

Faceted browser can vary in design. For example, faceted browsers can be either directional or non-directional. In a directional faceted browser the filtering of facets goes in one direction. Browsers such as mSpace and iTunes, for example, support directional facets ([schraefel et al. \(2005\)](#)). Additionally, faceted browsers have been researched extensively to improve performance in navigation and exploration. In directional browsers, for example, techniques like backward highlighting have been researched to study the effects of showing alternative paths that users could have taken to the items currently displayed ([Wilson et al. \(2008\)](#)). Other examples include displaying the number of items in each facet which is a contribution brought over from query preview interfaces

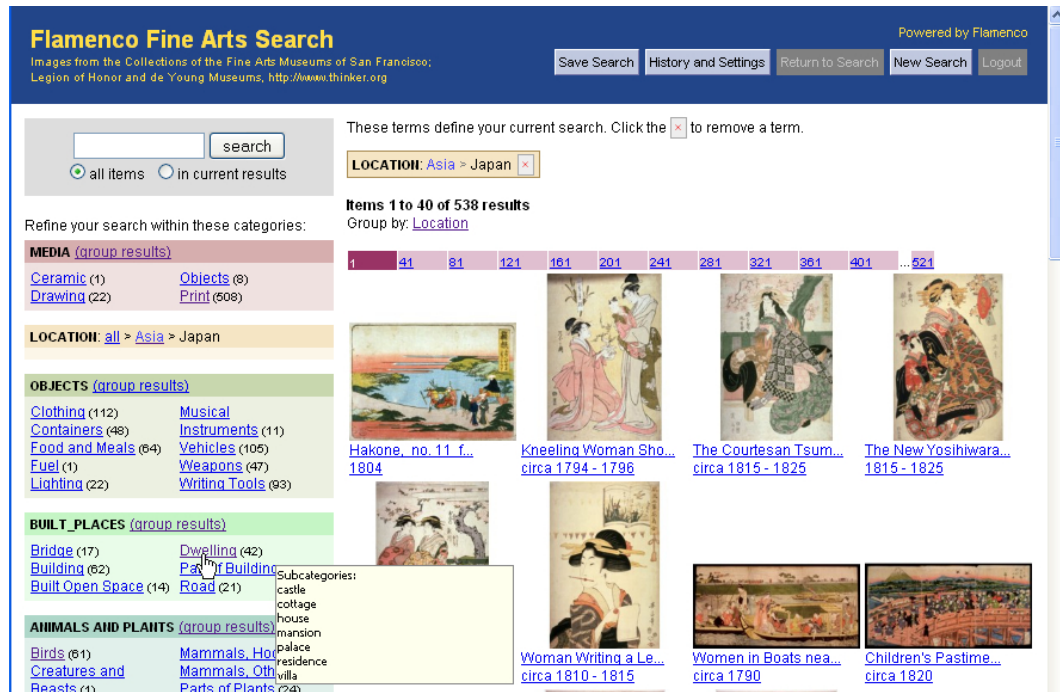


Figure 2.2: The Flamenco faceted browser.

(Plaisant et al. (1999)). Displaying the number of items presents users with a summarisation of the distribution of data and can additionally help identify potentially erroneous choices. Other approaches to displaying cardinality include representing them with a visualisation. For example, the Relation Browser and the Elastic Lists browser both use visualisations instead of using a textual representation to represent the distribution of items across facets (Zhang and Marchionini (2005); Stefaner and Muller (2007)). Representing facet selections can also vary depending of the data presently under inspection in the faceted browser. For example, studied different selection widgets with embedded visualisations for navigating through different types of content (Willett et al. (2007)). Since faceted browsers immediately expose the metadata associated with an information space they have been extensively studied in the context of exploratory search - a search scenario where the user is unfamiliar with a domain (Marchionini and White (2008)). A survey of different approaches to faceted browsing including visual design, interaction and structural design can be found in (Clarkson et al. (2009)).

2.1.1.3 Visualisations and Multiple Representations

The same information can show different insights when presented differently (Figure 2.3). For example, a map showing the major impact sites of a natural disaster might offer very different insights to only listing the sites in a tabular format. Charts are frequently used for comparison of quantitative aspects of data. The usefulness of data representation is often dependent on the users tasks. For example the iTunes³ music player allows people

³<http://www.apple.com/itunes/>

to browse through their music collections in several views, including a list of songs view in a tabular format or an album cover view format showing albums as thumbnails. In searching for a particular album, the latter would be a more appropriate representation to work with. A list of alphabetically ordered songs, on the other hand, might be a more appropriate choice when searching for a particular song. Data representation can also influence how data is selected for filtering. For example, geographic filtering would be facilitated better if a user can directly draw a region on a map, rather than specify a region by adding quantitative information about polygons. Likewise, events data might be more useful to access and select through a calendar application. Selection of a specific time period can be supported better by working directly on a timeline.

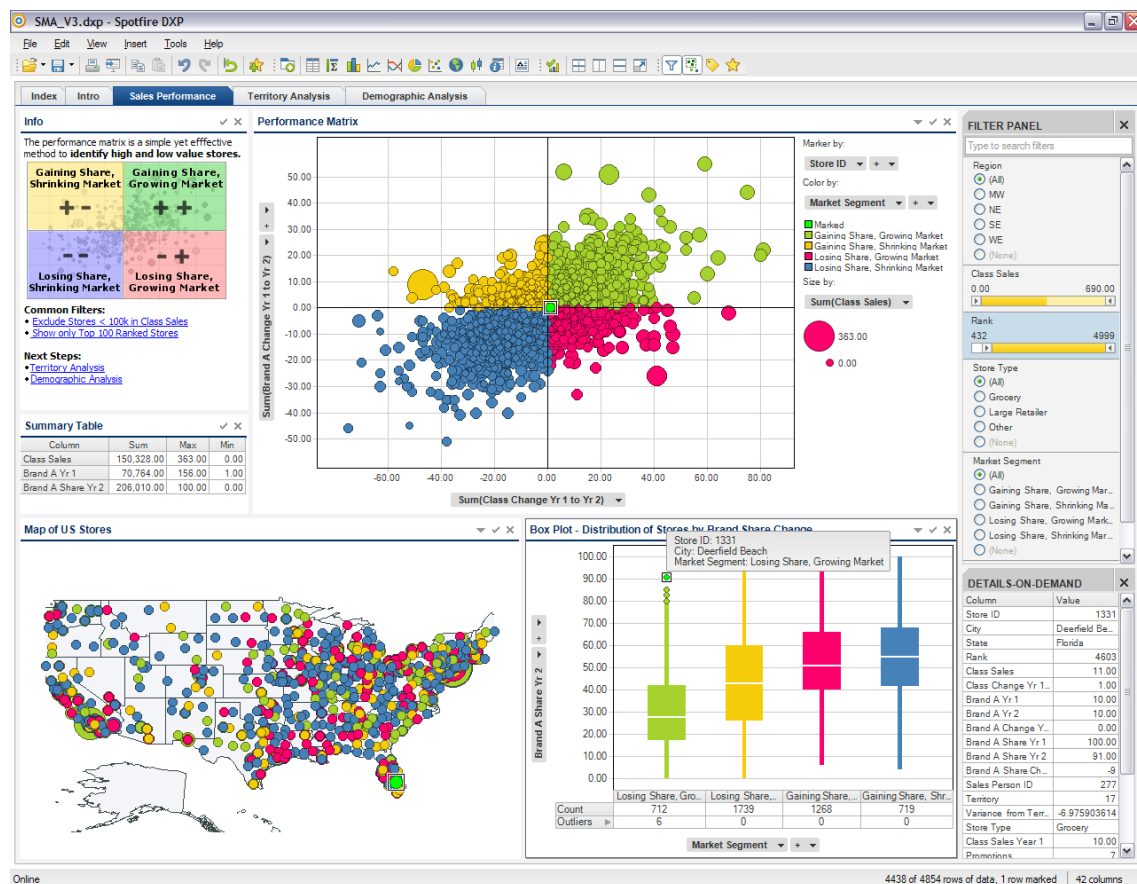


Figure 2.3: The Spotfire tool using different visualisations over the same source of structured data (Tanin et al. (1997)).

2.1.1.4 Data-centric operations

In addition to supporting structured search, information in structured format allows users to perform data-centric operations, such as sorting and filtering. For example, cases where the attributes of the information items have an understandable sequential order allows them to be arranged in a particular order. For example, a set of mobile phones can be sorted based on price (for example from most expensive to least expensive). For

nominal attributes which do not have a meaningful order, it is common to provide users with a filtering option that can be used in conjunction with sorting. Filtering is thus used to eliminate some records to help focus on categories of interest. Faceted browsing is considered to be a particular form of filtering.

Filtering and sorting are common features on many data-intensive websites and have been extensively used in a variety of other tools, such as personal information management tools and e-mail clients. [Cutrell et al. \(2006\)](#), for example, did a longitudinal study of the use of the Phlat personal information management system with more than 200 people over an 8-month period. They found that 47% of all queries used some kind of filter, one third of all searches that used filters used more than one filter, and 17% of searches used only filters, with no initial query term at all.

2.2 Structured Data on the Web

In the previous section, this Chapter discussed how having structured data improves finding, exploring, and interacting with information. In this section, the Chapter turns its attention specifically to structured information and data on the Web. The Web has undeniably been the single most significant medium in providing users with unprecedented access to information. While the Web is considered a global distributed repository of documents or web applications, it inherently draws most of its information resources from structured repositories (most commonly databases) of the individual Web publishers. The raw data content of these repositories is currently inaccessible directly on the Web (Figure 2.4) - the content is only accessible through the tailored experiences of the web site. Initiatives, such as the Linked Data, attempt to surface up raw data on the Web and advocate for publishing of structured data on the Web based on Web principles. In effect, the Linked Data movement promotes creating a global repository of structured information, one that can be queried, searched and the data can be reused in any number of applications. In this section, this thesis discusses sources of structured data on the Web and the technologies and principles of Linked Data publishing.

2.2.1 Sources of Structured Data

The amount of structured data on the Web today vastly outnumbers the information contained in static documents. Structured information is published throughout the use of database technologies; it can be user contributed through social media, embedded into documents or created into structured format out of unstructured documents. The following sections examine the most common sources of structured data on the Web today.

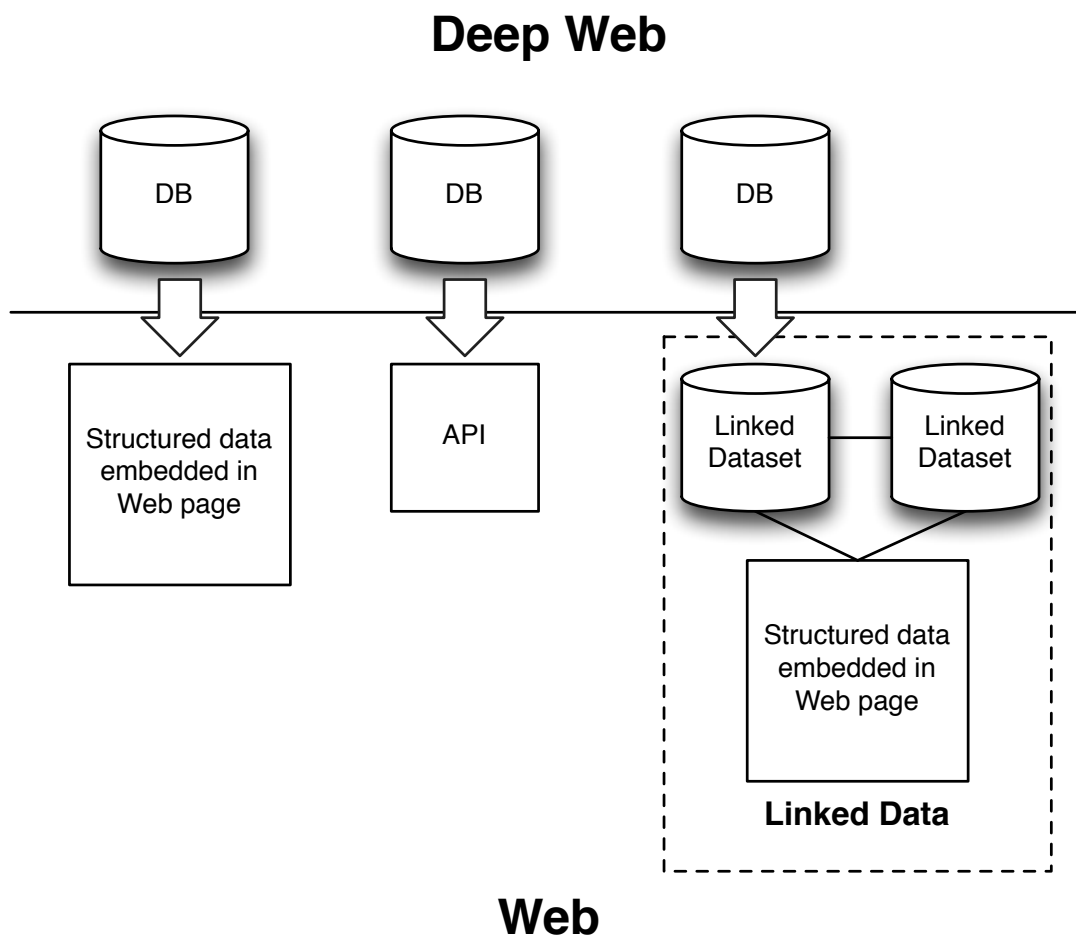


Figure 2.4: Different forms and uses of structured data on the Web.

2.2.1.1 Use of Data Technologies on the Web

The early Web was an entirely document-based enterprise; HTML documents embedded the information to be presented to users and the documents were placed as static artefacts on the Web. The information contained in the documents, however, rarely contained any structure or metadata about the content of the document; rather the only structure in the document was how the document was to be rendered in a browser. Quite soon after its initial inception, however, dynamically generated documents were required to handle increasing amounts of data. To facilitate this demand, websites were increasingly being joined with backend databases; the databases provided scalable data storage and ability to perform data-intensive operations, while the web document acted as the lens through which information was presented to users. Database technologies, complemented with server-side templating technologies such as ASP⁴, PHP⁵, and JSP⁶,

⁴<http://www.asp.net/>

⁵<http://www.php.net/>

⁶<http://www.oracle.com/technetwork/java/javaee/jsp/index.html>

allows today's publishers to publish large quantities of information efficiently and offer data-centric features such as sorting, searching, and filtering.

Using databased-powered websites promotes the paradigm of separation of data-storage and data-oriented tasks from the logic of the application and presentation of content. Unfortunately, the structured information which has already been provided gets reverted to unstructured information once it reaches the browser. Until now, the choice has been economically justified - unless other users could reuse the information, adding additional structure to the web page provides additional overhead and yet another concern to the website publisher. Additionally, in order to control access to what sort of data is reusable most data-intensive websites use APIs, which provides stricter access to structured data.

More recently, the demand for richer web applications have promoted populating the client side application with structured information. Data markup embedded into web documents is one example. Rich client development frameworks allow data to be stored into the client. The new specifications of HTML5, for example, implement a local storage feature that is a key-value pair storage on the browser. Despite these advances, it is still difficult to get raw data out of the information we see on Web pages. At present the only viable options are relying on a publisher's API or scraping the HTML for data. There are very few instances of interfaces such as Exhibit, which allow the data underlying the information presented on the website to be surfaced with a click of a button (Huynh et al. (2007b)).

2.2.1.2 User Contributed Content

The initial proposal for a Web of Documents, outlined by Tim Berners-Lee was originally conceived as a medium not only for consuming information but also as a collaborative space where users can contribute to the content of a website. The first Web browser⁷, for example, enabled link editing to any Web document accessible on the Web. Despite the popularity of the Web as a content consumption medium, it was not until the advent of Web 2.0 that casual users actually became active participants in generating content. Currently casual users contribute a large amount of the Web's content through blogs, wikis, social networks and micro-blogs. Most of the data is collected through contributed content, and stored into databases with predefined schemas. Several instances exist where users contribute structured data directly. One such example is perhaps Wikipedia⁸, where in addition to authoring articles, users can add content to Wikipedia's info-boxes which hold structured data about an article. The Wikidata Project⁹, aims to extend this functionality onto a specialised service (Vrandečić (2012)). Another similar example is Freebase¹⁰ Bollacker et al. (2008), which is an open database

⁷<http://www.w3.org/People/Berners-Lee/WorldWideWeb.html>

⁸<http://www.wikipedia.org/>

⁹<http://meta.wikimedia.org/wiki/Wikidata>

¹⁰<http://www.freebase.com/>

where users can contribute structured data in a similar way to how users contribute articles to Wikipedia.

2.2.1.3 Structuring Unstructured Data

The lack of structured information about content on Web sites has been the motivation behind several approaches to extract structure by using the document structure of webpages. Various algorithms exist to identify records from web pages; for example algorithms relying on partial tree alignments (Zhai and Liu (2005), Shen and Karger (2007)), tree edit distance (Reis et al. (2004)), and tabular structures (Lerman et al. (2004)). Depending on whether the user is involved in accurately identifying records, approaches to finding structure in web documents can range from completely supervised to unsupervised. End user tools that support structuring of web content can use the derived structure for different purposes; some tools use the derived structure to allow people to clip particular portions of a website for supporting information gathering tasks; other tools use identified structures to augment existing webpages with new functionalities. For example, tools like Hunter Gatherer allow users to select components out of webpages and add them to custom collections of content (schraefel et al. (2002)), while tools such as WebSummaries allow users to select or add specific metadata and save extraction patterns to be shared with other users (Dontcheva et al. (2006, 2008)). In other cases, tools such as Sifter serve to augment a Web site through adding facets from metadata extracted from websites by analysing its content (Huynh et al. (2006)). Piggy Bank, on the other hand, allows keeping a personalised collection of information through an automated extraction which can then be visualised and explored (Huynh et al. (2007a)). This allows browsing through the data with additional facets that might not be provided by the data publisher. Other approaches are not motivated by end user concerns but rather by being able to repurpose the data. The DPedia project, for example, is a database created by extracting data from Wikipedia info-boxes and mapping them to a predefined schema. The data gathered can then be repurposed in other applications (Auer et al. (2007)).

2.2.1.4 Page Markup

Web pages appear quite static to the outside world despite their dynamic generation; usually the databases that power them work in the background and are inaccessible to the outside world. Extracting structured data from unstructured documents uses heuristic algorithms and are thus prone to errors and often require user intervention. Another source of structured data in webpages comes from web publishers who provide some additional metadata mark-up to the content. Several services have recently emerged that have incentivised publishers to mark-up their websites with additional metadata. For example connecting websites content to Facebook allows users to share and support

particular content through its “Like” feature. Initiatives like schema.org¹¹, for example, allow for mark-up to be picked up and processed by search engines, which can render rich descriptions of websites once results are retrieved from a search (see Figure 2.5). Most of the mark-up is enabled through a number of data-markup languages such as Micro-data, Micro-formats and RDFa (Adida and Birbeck (2008)).

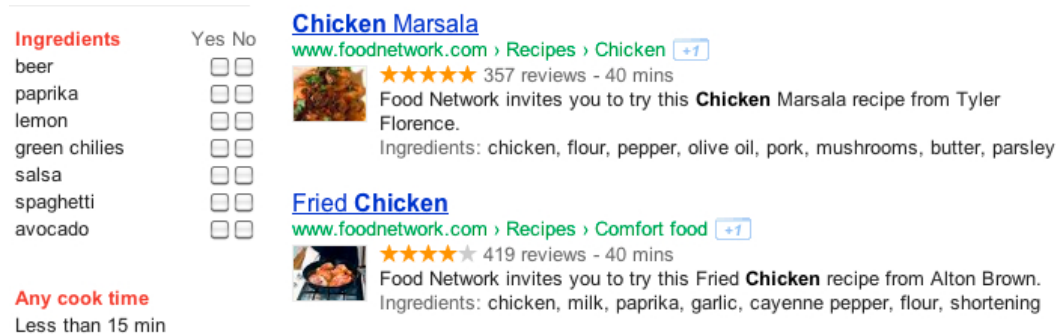


Figure 2.5: A Google search on recipes shows rich visual feedback due to structured data imbedded in web sites.

2.2.2 The Semantic Web

Information-oriented applications clearly benefit from using structured data by providing richer interactions through data-centric features. The tools in the examples shown in previous sections, however, suffer from one core limitation; each tool is a tailored experience, specifically designed to support only the data for which it was originally designed. In other words, those tools do not allow us to easily integrate and use several sources of information, nor do they make it easy to pick up the structured information and reuse it in another tool.

The Semantic Web is a proposal to make structured data more directly accessible on the Web. It proposes extending the Web to include the data published using common formats based on Web principles (Berners-Lee et al. (2001); Shadbolt et al. (2006)). The infrastructure of structured information would be published in addition to the Web of documents and applications, allowing web applications and developers to access data that is globally accessible on the Web. The Semantic Web vision advocates publishing structured information on the Web based on adopting a common stack of technologies (Figure 2.6). The core principles include representing structured information with a common data model, resources being uniquely identified with global, Web identifiers, the ability of these resources to refer to one another and describing the conceptual characteristics in commonly accepted languages. In effect, the Semantic Web promotes establishing a global distributed database of structured information sources to be globally accessible to applications. Unlike the document Web which is only accessible to human users, the original Semantic Web proposal envisions a machine readable layer

¹¹<http://schema.org>

over which applications can share and reuse data. The vision of the Semantic Web is summarised by [Berners-Lee et al. \(2001\)](#) in which they define it as:

“... an extension of the current Web in which information is given well-defined meaning, better enabling computer and people to work in cooperation.”

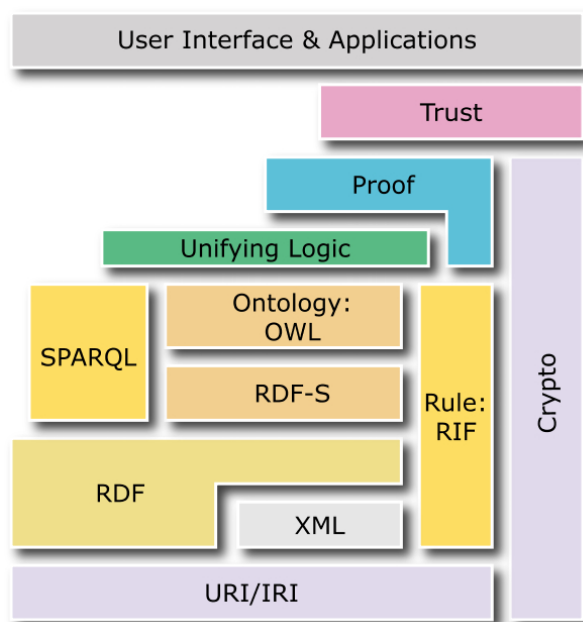


Figure 2.6: The Semantic Web technology layer cake.

2.2.2.1 The Web of Linked Data

The Semantic Web is a vision for a Web where intelligent programs or agents can operate over distributed data sources to either automate certain tasks or work together with end-users to accomplish tasks on demand. This scenario of a Web of highly interoperable information over which autonomous agents can interact with information requires enabling most of the technology stack shown in Figure 2.6. The basic foundations of the Semantic Web, however, start with publishing structured data using common standards and in a way that is compliant with the general architecture on the Web. Because of the many technical limitations of efficiently utilising the upper stack technologies and still evolving standards, the Linked Data initiative aims to utilise the lower stack of technologies as a way of publishing for the purposes of basic data integration and data repurposing. Often the terms Linked Data and Semantic Web are used interchangeably, however strictly speaking Linked Data is just a way of publishing data on the Web according to a number of principles outlined in [Berners-Lee \(2006\)](#). In the past five years, the amount of data published through Linked Data principles has risen steadily. Figure

2.7 shows available data sources and datasets published as Linked Data. Each node on the figure represents a dataset of Linked Data published either by a single data publisher or a data source - a community of publishers of particular type of data (for example FOAF [Brickley and Miller \(2010\)](#)). Each arc on the Figure is denoting that links exist between the data of the two data sources.

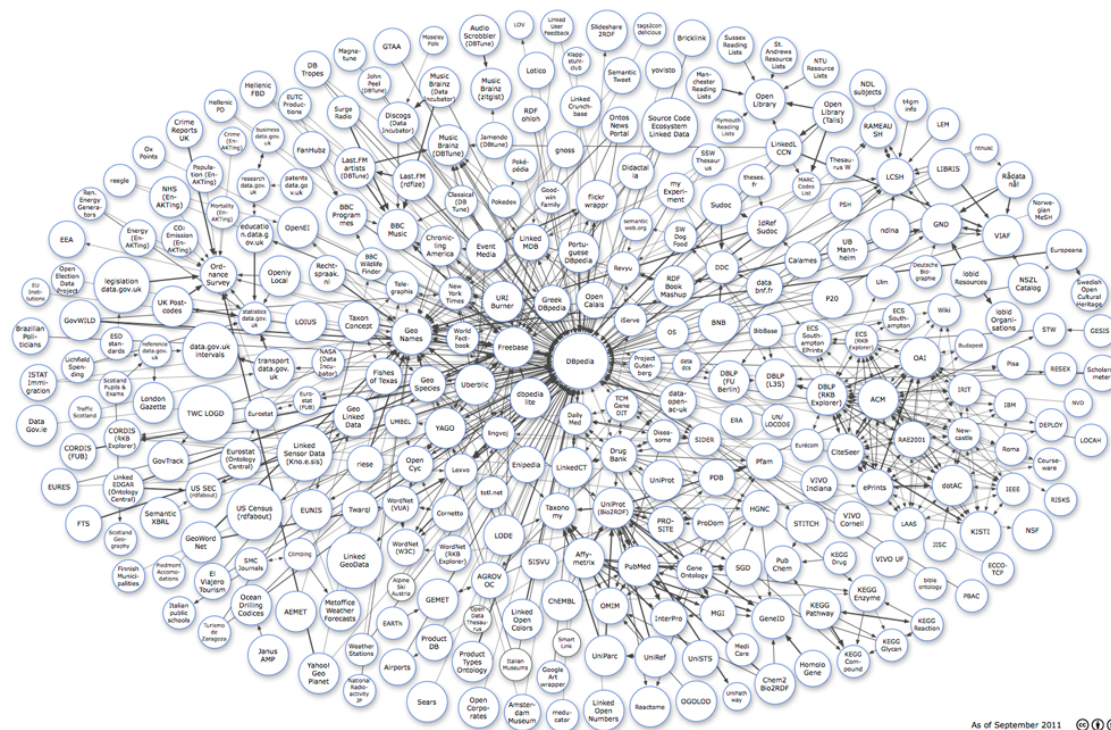


Figure 2.7: The Linked Open Data cloud shows datasets publicly available on the Web and the links they have to other datasets.

2.2.2.2 Concepts

Publishing Linked Data rests upon two basic technologies: first the notion of unique identification on the Web through URIs and an adoption of a common model for describing data through using RDF. In the following section we describe these two core concepts. Additionally, we describe concepts such as schemas and ontologies. Ontologies¹² are not a prerequisite to publishing Linked Data, although by publishing data one implicitly defines a schema. In this section we briefly discuss ontologies as a concept because frequently schema level information is published alongside the data (particularly through RDFs), and adopting vocabularies and schemas from other data publishers is a common occurrence on the Web of Data.

¹²By ontologies here I mean machine readable descriptions about the conceptual level description about the data.

Unique identification. Structured resources on the Web need global identification in a similar way to how webpages also need unique addressing to be accessed. Traditionally URIs have been used to uniquely refer to Web pages and these have been more commonly known as Unique Resource Locator (URL)¹³. When publishing Linked Data, the basic unit of publishing is a resource which is usually a real-world or abstract object - for example a resource might be a particular person, event, city, or perhaps a media type, such as a blog post, micro-note or status update. When publishing Linked Data, individual resources are identified through URIs, which in effect makes each resource a first-class citizen on the Web. The principles of Linked Data mandate that the identifiers of resources be HTTP URIs which means that they will then need to be resolvable through the HTTP protocol¹⁴. HTTP URIs enable information about the resource to be retrieved once resolved in a browser or requested by other applications through HTTP.

Common data model. The commonly adopted model for describing data on the Web is the *Resource Description Framework* (RDF)¹⁵, a graph based model for describing data. RDF describes data in terms of statements (also known as triples) where each statement is composed of a subject, a predicate and an object. For example, Figure 2.8a shows a simple statement saying that the book, “Lord of the Rings”, is authored by “J.R.R Tolkien”. In RDF, a resource and a property are identified through URIs i.e. they are both resources, while an object can be either another resource in which case it is again a URI, or can be a *literal* (Figure 2.8b), a string-like value that denotes some attribute of the resource that is not a resource itself - for example a book’s title, price, or number of pages.

```
(a) <http://example.com/book/Lord_of_the_rings> <http://example.com/bookauthor> <http://example.com/author/JRRTolkien>
(b) <http://example.com/author/JRRTolkien> <http://xmlns.com/foaf/0.1/name> "J.R.R. Tolkien"
```

Figure 2.8: Information about the book “Lord of the Rings” in RDF.

On the Web the most granular unit of information is a Web document, which can contain unstructured information about multiple topics. RDF, on the other hand, provides a data model, that can be used to describe data at arbitrary levels of granularity. For example, RDF can be used to model data in relation to real world objects or it can be used to model data in an described data model (e.g. a tabular representation of data in RDF). A Web document is a media type created for human consumption - the information is joined with a presentation template created by the publisher. RDF, on the other hand, is just a data-model, one that can be serialised, i.e. represented using a number of languages, such as RDF/XML and N3. By publishing data using common standards and principles, the aim of the Linked Data project is to create a

¹³<http://www.w3.org/Addressing/>

¹⁴<http://www.w3.org/Protocols/>

¹⁵<http://www.w3.org/TR/rdf-primer/>

global information space of structured information where data is inter-linked in a similar way to how the Web is a global space of inter-linked documents.

Ontologies and schemas Ontologies describe conceptual level information about the domain of some structured data. Gruber (1993) describes ontology specification of a conceptualisation. Ontologies capture essential information including what type of data is contained, what are the relationships between entities in the data, and any specific rules. For example, if our domain is publishing data about books, the ontology would inform that instances can be of type books and authors, that the books can have one or several authors, that the information about books contains information regarding the number of pages, references etc. Ontologies can be diverse with respect to the level of description they provide. An ontology can range from simple representation of knowledge such as taxonomies or thesauri, to more complex knowledge and more formal representations (McGuinness (2003)). Similar to how the data itself is described in structured format, an ontology can be described formally using languages such as RDFs¹⁶ and OWL¹⁷. The former is generally used to formalise basic information about a domain, such as classes, properties, property domains and ranges, while the set of OWL languages provide greater expressivity which includes things such as specifying axioms, cardinality, equivalences etc. We should note that ontologies have a different meaning in the context of the Semantic Web; by definition an ontology is a conceptualisation of a domain; however in the context of the Semantic Web an ontology is the expression of the domain in languages such as OWL and should be considered more related to a database schema.

2.3 Data-centric Interfaces over Linked Data

The ultimate aim of the Semantic Web is one in which software agents can perform various tasks over Linked Data, including organising and enabling more efficient exploration of information than traditional models based on keyword search. This thesis, however, attempts to answer a more modest problem: can Linked Data, as a medium of raw structured information on the Web, be directly accessed by non-technical users in order to solve information tasks that require combining several data sources? This problem can be broken down into several sub-problems. First, what are the main challenges to developing generic tools that can provide access to Linked Data on demand? How can potential users triage a large and complex set of data in a dataset and figure out if the data contained in a dataset can be used to answer a particular question? How do we best represent data in machine readable format to facilitate certain tasks? Who are the users, and what are the potential use-cases? What biases and constraints do they bring to a system?

¹⁶<http://www.w3.org/TR/rdf-schema/>

¹⁷<http://www.w3.org/TR/owl-features/>

This thesis is primarily concerned with the afore-mentioned interaction challenges. In addition to the interaction challenges, accessing Linked Data in a truly distribute way requires solving a number of technical, back-end challenges. Providing solutions to these problems, however, is out of the scope of this thesis since these problems present general problems to consuming Linked Data and adoption of the Semantic Web, and thus do not only relate to end-user interaction. In the following section, this thesis argues why end-users would benefit from having access to Linked Data on the Web, and presents the key challenges to enabling such interfaces. It also reviews related work in a number of active areas of research around data-centric interfaces. It also briefly presents some of the back-end issues and discusses the limit of potential end-user contributions to solving some of these challenges.

2.3.1 Defining Data-centric Interfaces for End Users

Interfaces over Linked Data come in various forms and for different purposes. In most cases, it is likely that interfaces over Linked Data would only replace databases as the data access layer and provide the same tools that are built over databases today. Thus, in order to distinguish these instances of interfaces from *data-centric* interfaces we adopt, the following criteria for defining a data-centric interface:

- Interfaces that are neither assembled over a particular dataset nor mandate an input from a particular type of data, but rather access data on demand.
- Interactions that facilitate finding, combining and querying multiple sources of data using the links in a graph of data.
- Interactions that allow sense-making of graphs of data for the purposes of meeting data-centric needs.

The most common example of data-centric interactions over Linked Data is generic data browsers. Generic data browsers use an analogy of a Web browser; however, instead of browsing documents, it supports browsing data.

2.3.2 The Case for Supporting Data-centric Interactions over Linked Data

Throughout this Chapter, various examples about ways of how structured data enables new exploration features were given. The examples listed so far, however, exhibit one common pattern - all of the examples are tools that were custom built to enable a particular set of interactions over some predefined data. In other words, we are allowed to interact with the data only in ways provided by the publisher or application developer.

While such interfaces promote the case of using structured information, they restrict us from either reusing the data for other purposes or using it together with other data sources. Take for example a faceted browser used to browse over data about mobile phones: the browser might offer finding phones through common facets: price, manufacturer, phone model. However, what if we wanted to filter through a property not supplied by the browser e.g. battery life? Or what if we wanted to compare prices across multiple online shops, and perhaps render a nice bar chart visualisation to convey the comparisons? At best we would have to go through each phone we found on the first site and do a search for each phone on another online site. Additionally we would be restrained by the inability to quickly extract these information resources - rather we would have to resort to copying and pasting text snippets and aggregate them in a spreadsheet to generate a desired visualisation. By having data represented in common formats and integrated through linkages as Linked Data, we can begin to envision interfaces which can gather, and represent data about related topics in a way that suits the needs and requirements of the user.

2.3.3 Challenges of Developing Data-centric Tools for Exploring Linked Data

Challenges in delivering end user data interfaces over Linked Data include both back-end challenges and front-end challenges. Back-end challenges typically address problems of scalability and data heterogeneity. Front end challenges include having sensible representations of data and interactions that allow users to find and interact with data in ways that are usable and take advantage of the inherent linkages that exist between data sources. This section gives a brief overview of the key problems, both front end and back-end, in order to give readers a holistic view of the challenges of having fully fledged, generic data-centric tools over distributed data on the Web.

2.3.3.1 Front-end Challenges

Defining Sense-making over Linked Data

Publishing Linked Data is based on the notion that accessible data on the Web can be reused by other parties, typically technical users, such as developers, in developing new services. Data-centric tools, such as data browsers for casual users, explore the possibility of having nontechnical end-users, those without technical skills, to access and reuse this data for their own data-oriented information needs. Thus, one of the challenges of having data-centric browsers is to allow sense-making over unfamiliar datasets containing raw data published on the Web.

[Russell et al. \(1993\)](#) defines sense-making as the process of searching for a representation and encoding data in the representation to meet a particular information need. [Pirolli](#)

and Card (2005) reinforces this idea by arguing that sense-making is the ability to represent data into a schema, whether internal or external which can aid analysis. Klein et al. (2003) data-frame theory views the sense-making phenomenon in a similar context. However, Linked Data on the Web is already structured data and published based on some schema which the data publisher has already used to publish the data. Thus the task of sense-making over Linked Data needs to be seen in another light; sense-making over Linked Data is the ability of end users to explore and find appropriate data sources combined with the ability to piece the various information sources in order to meet an information need. This challenge can be broken down further. First, what sort of data representations can the user handle? For example, some users can easily use more generic representations, such as tables and spreadsheets, while other, less data-savvy users need richer representations typically facilitated by custom build applications. Second, how do people use the links between resources in Linked Data to find and combine different data sources? Most of the generic data tools currently use navigation in the RDF graph to facilitate finding related data resources starting from a selected set of resources.

Presenting data Since RDF by principle promotes separation of data from presentation, finding suitable data representations for to the user is a significant challenge. Creating data-centric browsers involves decisions on how closely the representations of data in the browser relates to the underlying RDF data model. If a browser imposes a level of presentation that highly abstracts the underlying data, then there is a danger of losing its generic attribute. On the other hand, if the browser chooses to closely relate to the underlying model, the user experience can be confusing for casual users with no knowledge in data representation models and schemas. Commonly, generic browsers rely solely on the structure of the RDF data model, i.e. the notion that everything is a triple for representing data in the browser. A RDF resource is thus often used as the smallest piece of information that can be represented in a browser. This, however, assumes that all RDF resources can be considered as self-contained information resources that are self-describable. While this is usually the case with most data published as Linked Data, it cannot be assumed as a general case. Perhaps standardisation in publishing - for example adopting minimal conventions (e.g. all resources must have a label property used for rendering data in human readable formats) and use of a small set of common vocabularies (e.g. labels are always described with the `rdfs:label` property) can give more structure upon which generic data browsers can build on.

2.3.3.2 Back-end Challenges

Scalability Depending on various implementations, Linked Data browsers can process either single or a relatively few RDF resources simultaneously, or can choose to enable browsing over large volumes of data. This typically depends on the interaction that the Linked Data browser intends to support. For example, earlier data browsers mimicked

how normal Web browsers worked - browsing was typically done one RDF resource at a time. Dereferencing a single RDF resource into a browser is not more computationally or network intensive than rendering a Web page in a browser. Using dereferencing, however, would make browsers that support browsing through large volumes of data unresponsive. For example, it would be difficult to support faceted browsing over more than a few hundred items without the use of a database. To facilitate such data intensive features, large publishers of Linked Data rely on centralised triple stores with query capabilities such as SPARQL to store and query graph data ([Prudhommeaux and Seaborne \(2008\)](#)). Unfortunately, this once again results in fragmented Linked Data sources on the Web, which require back-end support to be integrated into a single store where they can be queried jointly. While research into federated queries is an active topic in the Linked Data community (see [Hartig et al. \(2009\)](#)), these frameworks do not yet provide execution of queries with response times required to match the necessary responsiveness needed for a data-centric user interface.

Co-reference One of the core principles of the Semantic Web is to use URIs to identify real-world objects. However, different data publishers might use different URIs to identify the same object. For example, the two URIs `http://cities.com/paris` and `http://capitals.com/paris` can be used to identify the same thing - in this case the city of Paris. In the early days of the Semantic Web it was advocated to use the same URIs to identify same resources across different data publishers. Since the initial Semantic Web proposal, however, this approach was seen as the biggest scalability impediment for a world-scale data publishing effort. Additionally, as publishing data using the Linked Data principles began to emerge, requiring URIs to be resolvable inevitably meant that only the publisher of a particular URI can make statements about the object that the URI identifies. The Linked Data community has thus embraced this heterogeneity and the use of dereferencable URIs actually now deterred publishers from reusing URIs for identifying concepts ([Booth \(2009\)](#)). The problem of using different identifiers to identify concepts is known as the co-reference problem. A common practice nowadays is for publishers to add equivalence links between instances identifying same concepts. In terms of consuming data, however, this adds to the responsibility of dealing with this problem on the part of the application consuming the data.

Heterogeneity

Publishing data on the Semantic Web mandates using RDF as a common model for publishing data. However, using a common model for publishing data does not insure that two resources of the same concept follow the same structure. This problem is known as structural heterogeneity. Structural heterogeneity arises from the use of different ontologies for structuring data. For example, a data publisher might choose not to reuse an existing ontology like FOAF for describing people, but rather to create a custom ontology to suit the particular application needs. The result means that any application wanting to blend data from publishers that use and do not use FOAF needs to align

the ontologies (i.e. provide mappings) of both resources to present a unified view over the data. This creates problems for generic browsers, since most of them are only aware of the common data model (subject, object, and property) and need additional knowledge to jointly represent resources that are structured using different ontologies. Alternatively, generic data browsers can choose to expose data heterogeneity to users and support them in aligning the heterogeneous resources themselves (e.g. Potluck [Huynh et al. \(2008\)](#)). Data alignment, however, is a laborious job even with good support, and is unlikely to be picked up by casual users.

2.4 Areas Related to End-user Data-centric Interfaces

Research relating to data-centric interactions for non-technical users is presented in several communities. In the database community, for example, a number of Visual Query Languages aimed at improving database management by replacing command-line SQL commands with graphical user interfaces. With the arrival of the Web, and in the absence of structured data on Web pages, mashup tools aimed at helping to structure data directly from Web pages. The scraped data could be used to support information-gathering tasks or analysing information from several sources. Some mashups are also designed as end-user programming tools, allowing people to combine or execute actions over sources of structured information such as RSS or ATOM feeds and APIs.

2.4.1 Visual Query Languages

Since RDF is just another data-representation model, browsers designed to navigate graphs of RDF data have similar concepts to visual query interfaces that have been developed in the years following the wide adoption of relational databases (e.g. [Derthick et al. \(1997\)](#), [Azmoodeh and Du \(1989\)](#), [Zloof \(1977\)](#)). A comprehensive survey and classification of relational database visual query tools is given in [Catarci et al. \(1997\)](#). While some interaction principles are common with existing data browsers, there are several notable differences. First, most of these interfaces were designed to be used in a database with a limited dataset and not part of much wider, Web-scale datasets in mind. Thus, these interfaces do not have to account for large numbers of relationships and serendipitous discovery of data. Second, many Visual Query Languages were designed with the intention of providing better administration and management of databases, rather than to provide and empower the end-user for the purposes of sense-making. Some examples that have tried to marry aspects of visual query languages with direct-manipulation visualisation tools for the purposes of analysing data include Visage (Figure 2.9) and SnapTogether ([Derthick et al. \(1997\)](#); [North and Shneiderman \(2000\)](#)).

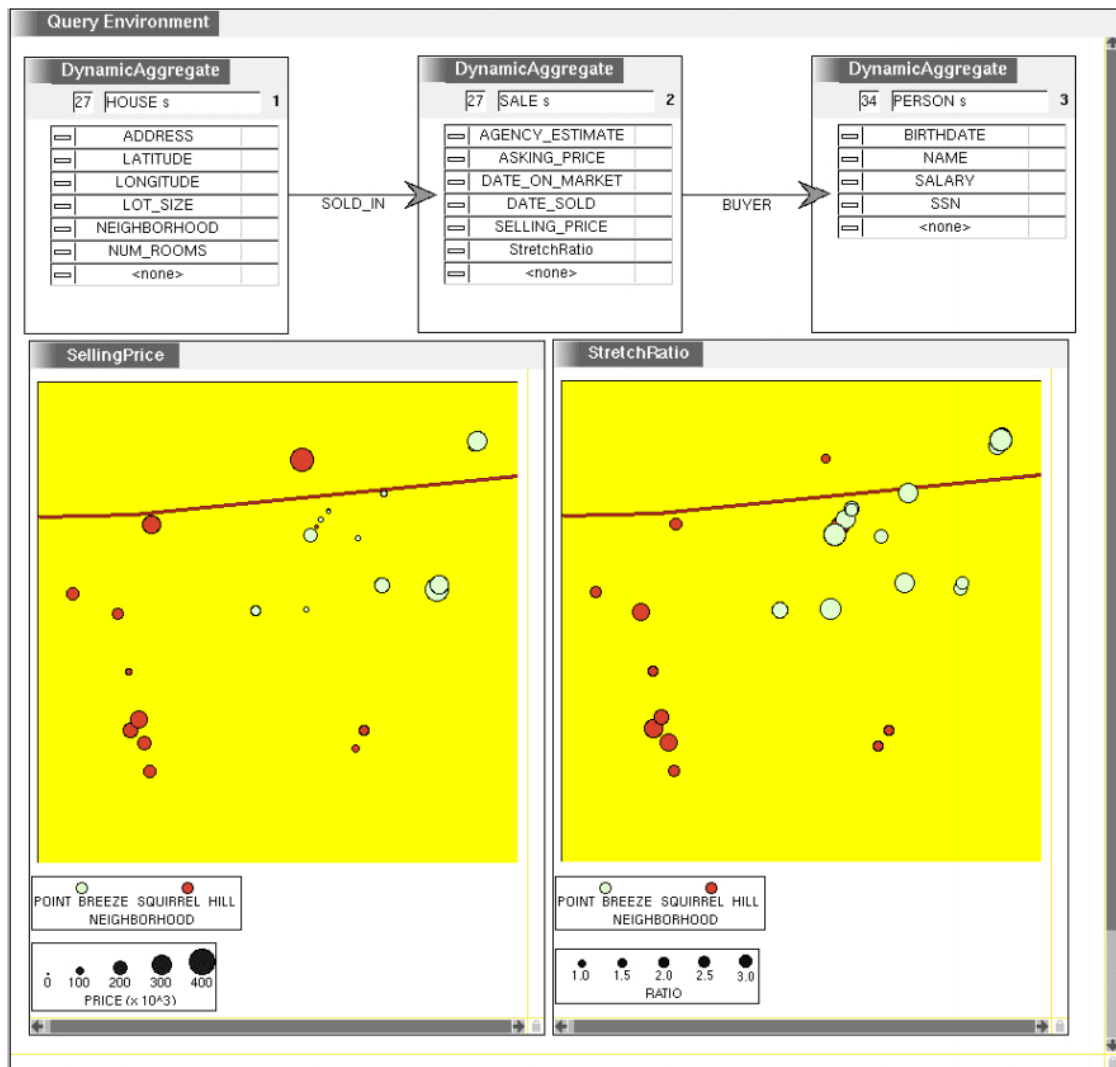


Figure 2.9: The Visage tool integrating visualisations with a visual query language (Derthick et al. (1997)).

2.4.2 End User Mashup Tools

Recently, a popularised method of publishing content on the Web is creating mashups i.e. publishing aggregated content from several sources into a coherent representation. While mashups have sprung up all over the Web (e.g. an RSS reader can be considered a mashup), most of these are tailored made Web resources developed by programmers and offered to end users with little or no flexibility for adding new sources or representing existing resources in multiple ways. Mashup interfaces for end users have been researched in both the context of mixing up web content as well as for mixing raw data on the Web.

The lack of structured data on the Web, and inability to organise content on a more granular level than web pages, provides the motivation for many end-user mashup tools. For example, WebSummaries is a tool that allows people to specify patterns in data-rich websites and create collections of structured information (Figure 2.10) (Dontcheva

et al. (2006, 2008)). The Intel mash maker also searches and extracts structured data out of multiple Web sites that can then be visualised and represented in multiple ways (Ennals et al. (2007)). Reform allows programmer-contributed widgets and browser extensions to be used by the end-user as a way of augmenting and viewing information on websites in different ways. Some approaches use unsupervised ways of structuring unstructured content (Toomim et al. (2009)). PiggyBank, for example, allows data to be automatically extracted from multiple pages to form collections that can then be browsed using a faceted interface (Huynh et al. (2007a)). Miro also uses a sophisticated data detection method that matches the semantic context of a Web page to potential user goals (Faaborg and Lieberman (2006)). TX2 uses connections found from gathering and mining interactions of users with web forms to integrate the search results from multiple pages by context in a single results page (Bigham et al. (2009)).

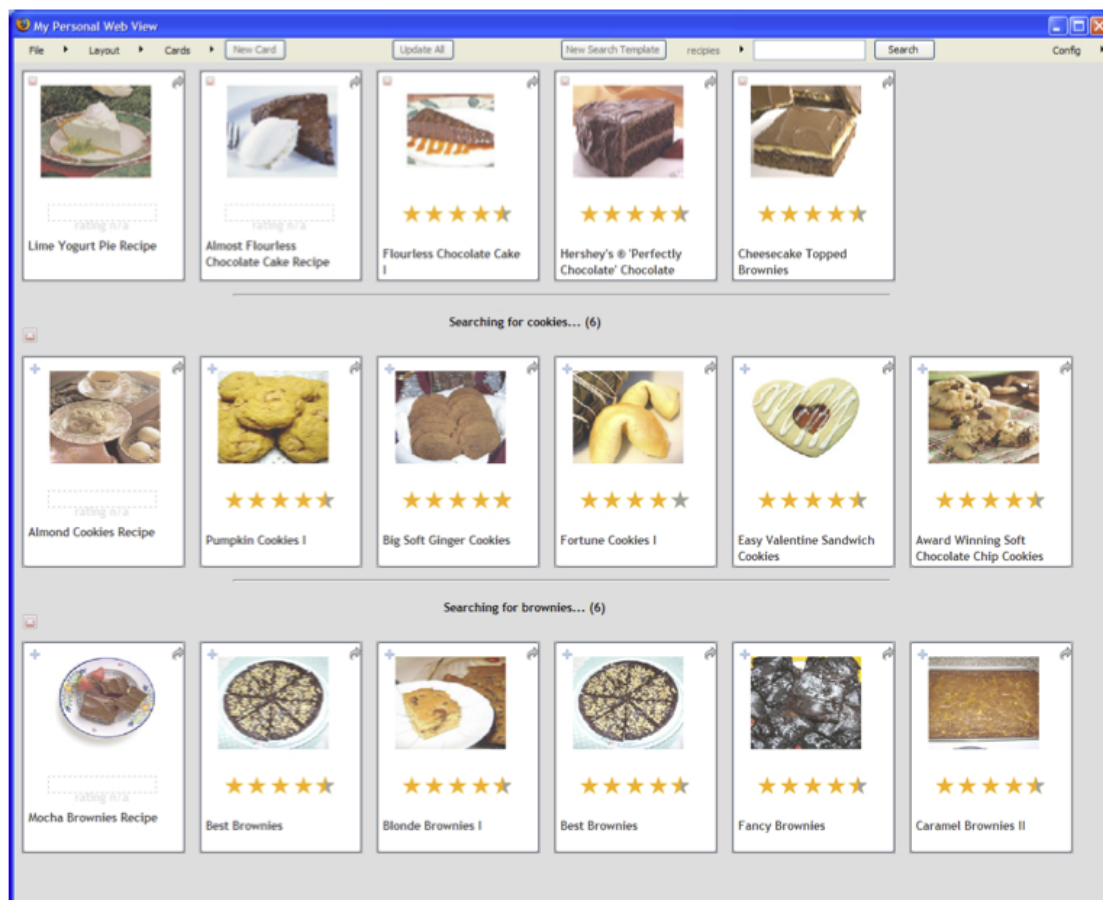


Figure 2.10: The WebSummaries tool Dontcheva et al. (2006).

There are also many mashup tools that allow end users to create mashups outside web browsing activities. Systems such as Yahoo Pipes¹⁸ (Figure 2.11), Marmite Wong and

¹⁸<http://pipes.yahoo.com/pipes/>

Hong (2007), Microsoft Popfly¹⁹, IBM QEDWiki²⁰, and Anthracite²¹ support combining data in an application-like interface using drag-and-drop interactions that graphically chart data flows and actions. The end-user programming method of these interfaces, however, hasn't been widely adopted despite being around for several years. For example, two of the three major tech companies offering end user mashup editors have discontinued these projects (Google's Mashup Editor²² and Microsoft's Popfly²³, the only one remaining being Yahoo Pipes).

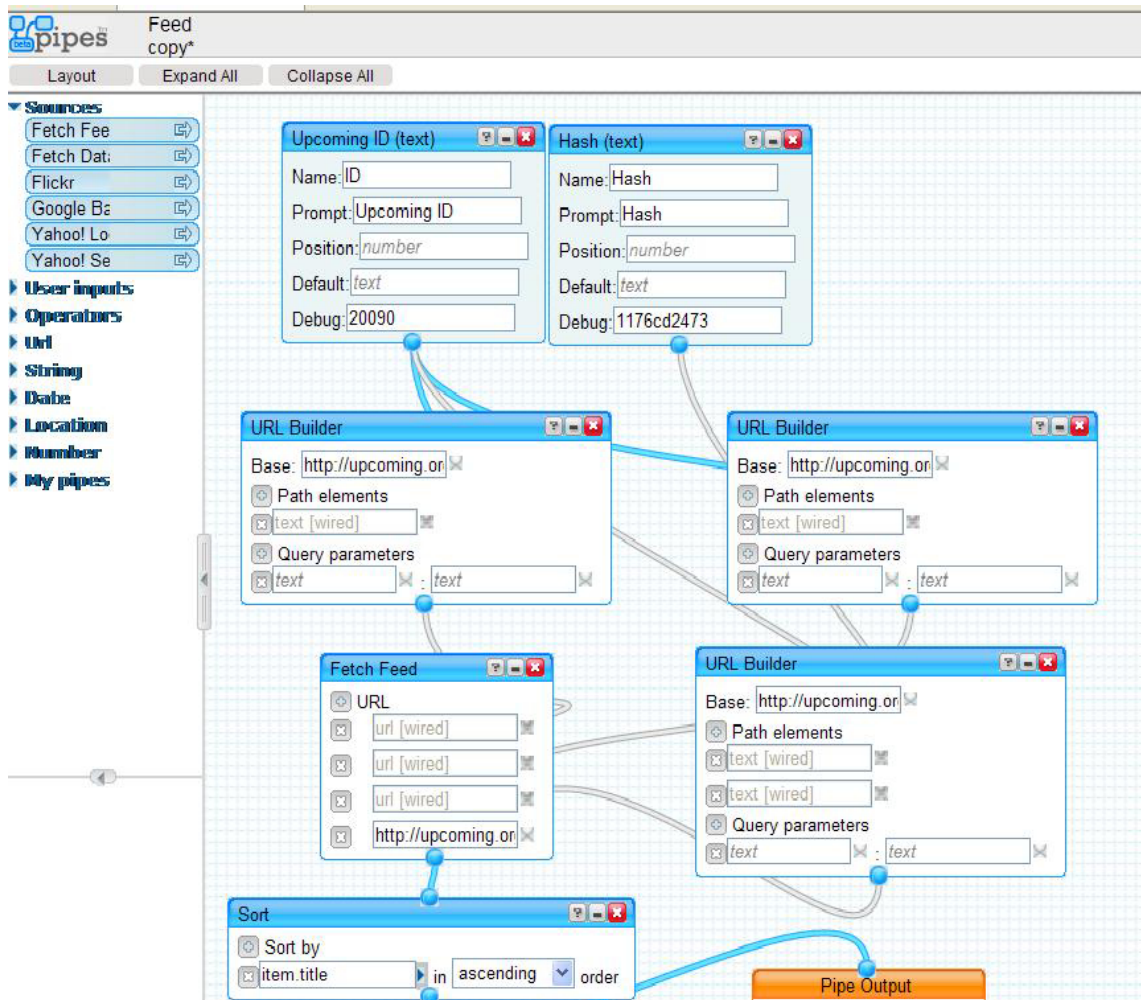


Figure 2.11: The YahooPipes user interface.

2.5 User Interfaces over Linked Data

In the previous sections, this thesis discussed sources of structured data on the Web, the aims and vision of the Semantic Web and it gave an overview of the general challenges

¹⁹http://en.wikipedia.org/wiki/Microsoft_Popfly

²⁰<http://www.youtube.com/watch?v=63qIq9t9Gqs>

²¹<http://www.metafy.com/products/anthracite/>

²²http://en.wikipedia.org/wiki/Google_Mashup_Editor

²³http://en.wikipedia.org/wiki/Microsoft_Popfly

and requirements for having usable data-centric interfaces over Linked Data. This section attempts to give a wide overview of the general area of interfaces over Linked Data designed for end users. The section touches upon several topics: various forms of interfaces, visualisations and interactions over Linked Data. Apart from giving an overview of Linked Data interfaces for end users, the core aim of this section is to present two fundamentally different approaches of applying interfaces for end users over Linked Data: the first, a *data-mapping* approach of applying interfaces over Linked Data that allows configurable interfaces to be added over predefined aggregation of Linked Data, and the second, data-centric interfaces that enable interactions over arbitrary Linked Data resources through navigation and browsing.

2.5.1 Representing Data

In general, two approaches can be used to represent data in a tool: (1) a direct visualisation approach which requires no configuration or requires no representational information in addition to the data, and (2) representing data through representational information, such as templates. A survey of various ways of visualising Linked Data in tools is given by [Dadzie and Rowe \(2011\)](#).

2.5.1.1 Direct Visualisation Approaches

Since RDF is essentially a graph of inter-linked resources, early Semantic Web interfaces used this inherent structure to present data in browsers as network graph visualisations. From a purely visualisation standpoint, some studies (typically in the domain of social networks) have shown utility of graph visualisations; however they usually tend to be utilised in specific circumstances. Graph-based visualisations can be useful for understanding data only when the number of nodes and edges is relatively small, for example when visualising your personal social graph ([Heer and Boyd \(2005\)](#)). Uses of large-scale visualisations of network graphs have been used showing certain aggregate attributes of data, e.g. showing density of connections between things in data and these usually tend to be used in conjunction with other visualisation aids such as colour, node arrangement and edge length and statistical information to make clusters visible ([Perer and Shneiderman \(2008\)](#)).

Graph visualisations for browsing data are typically found in interfaces which include authoring capabilities, both on the instance and ontology level. Examples of interfaces offering graph visualisation are IsaViz²⁴ and Fenfire ([Hastrup et al. \(2008\)](#)). Many ontology authoring tools such as Protege²⁵ typically use graph visualisations to convey complex ontology domain and numerous approaches have been proposed to visualise

²⁴<http://www.w3.org/2001/11/IsaViz/>

²⁵<http://protege.stanford.edu/>.

and browse ontologies at scale (Xiong et al. (2006); Motta et al. (2011); Howse et al. (2011)). Ontology visualisation, however, is appropriate for users that are knowledgeable in semantic technologies and data publishers rather than end users. m.c. schraefel and Karger (2006) argue that graph visualisations of instance level data do not offer any extra affordances for casual users and rather expose them unnecessarily to the complexities of the underlying data model. As an analogy, they point out that even though the Web itself represents a graph, it is never represented to the user as such.

Another approach of directly representing RDF data is to represent each RDF resource as a Web page where the properties linking from and linking to the particular resources are represented as hyperlinks. This approach has been adopted by early generic browsers, such as Disco²⁶, and is also an adopted paradigm for Linked Data publishers who usually resort to this particular type of representation when a URI is dereferenced from a browser. These representations are useful for technically oriented consumers of Linked Data, but are of little use when richer data-oriented interactions are required.

2.5.1.2 Use of Lenses and Templates

Machine readable data should never be surfaced up to the user, so most data-centric browsers utilise literal values in the RDF which are used to represent data to users. However, data can be shown in many different ways; for example each single RDF resource can be accompanied with a visual representation, or another example would be multiple RDF resources sharing common properties to be bundled up in order to show an appropriate aggregate representation. In order to facilitate multiple representations, reusable constructs such as *lenses* and other forms of templating (commonly used in server side powered webpages) have been used to capture knowledge about how RDF should be presented in an interface. An example (Figure 2.12) of a simple snippet of data represented through two different lenses illustrates this concept. The example shows data about UK local regions by applying two different lenses. The first lens is applied to each individual region which can be used to show a list of regions alongside information about their respective area size and pollution level. The second lens can take data from several regions, compute the total pollution relative to the area of each region and display them as an aggregate intensity map. Because the principle behind RDF by design advocates strict separation of knowledge and presentation, representation is usually left to the interface consuming the data.

Deploying representations of data can be either contributed by the developer of an interface over Linked Data or by the users which use the interface. Fresnel lenses, for example are encoded in RDF, and specify which portions of a RDF need to be selected for a given resource or resources of which have common attributes (Pietriga et al. (2006)). The Template Attribute Language (TAL) for RDF offers lightweight descriptions for

²⁶<http://www4.wiwiiss.fu-berlin.de/bizer/ng4j/disco/>

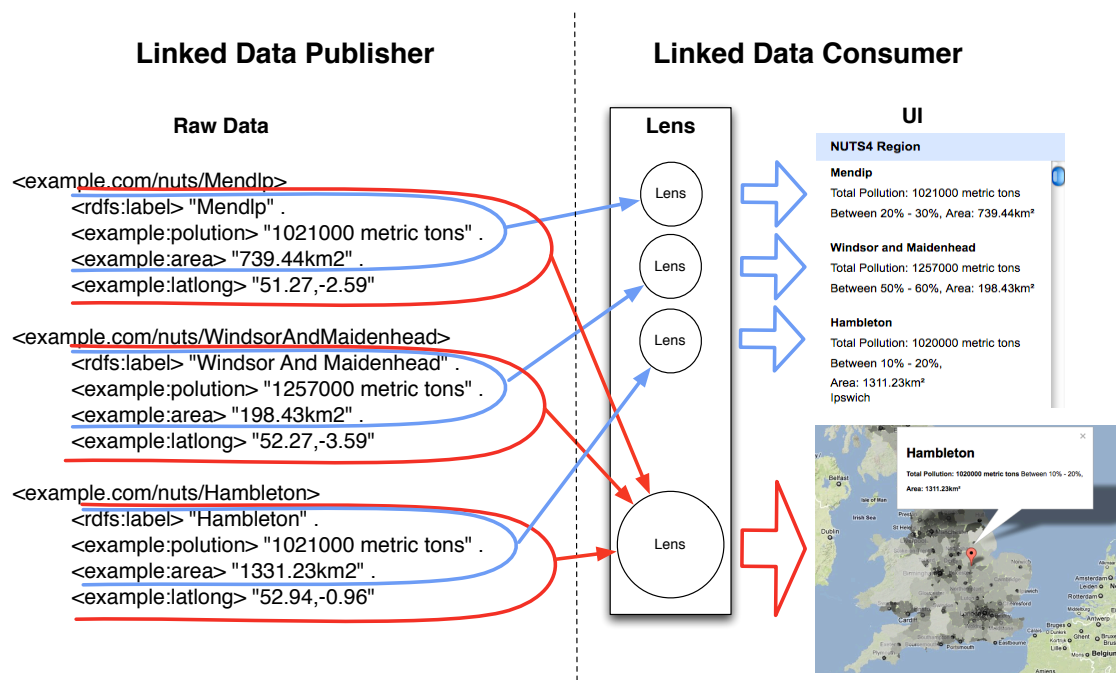


Figure 2.12: Use of lenses to display RDF data. Lens can be deployed per resource or can be used to show aggregate views of multiple RDF resources.

generating XML (typically XHTML) and textual representation of resources out of RDF (Champin (2009)). These approaches are typically used by developers of interfaces over Linked Data. Other approaches, such as SemLens allow displays of data in more raw and generic formats, such as lists and tables, to be selected and visualised through the use of widgets that can take data selections as input (Heim et al. (2011)). Dido allows users to add new visualisations and structured information about the lens through WYSIWYG interface (Karger et al. (2009)).

Concepts such as lenses are applicable to interfaces that can be configured to finite data sources. Because a dataset is finite, a publisher of a user interface can provide lenses for all the data because the data source is known a priori. Generic interfaces, however, assume that they can directly access any raw data and thus try to display data through various basic heuristics - for example looking for common properties, such as labels, titles and literal values. Both have advantages and disadvantages; in the former case, the lack of any representational information can result in a poor user experience, but it can offer greater flexibility and customisation to users, who can create representations that fit their needs. The latter, on the other hand, can offer a more usable experience of data and display useful visualisations and representations. A balanced approach would offer both suggested rich views and the ability to access the raw data on demand, but interfaces with such capabilities are rare, especially for generic data browsers. One possible example are Exhibit and Dido which offer users the option of exporting the

data used by the interface, but both tools are still fundamentally interfaces that support browsing over finite datasets and do not allow browsing through arbitrary Linked Data.

2.5.2 Data-mapping Interfaces

When rich interactive browsing experiences are published on the Web, the data is usually collected and maintained by the same party that designs and develops the interface through which the users access the data. While modules and reusable components exist, the implementation is usually tailor-made for each data publisher. One of the advocated benefits of Linked Data is that applications can be reused and deployed faster and more efficiently due to data publishing, using common standards. In this direction, there have been a number of proposals for generalising browsing interfaces in a way that allows any data in RDF to be quickly mapped or configured to the components of the user interfaces. These *data-mapping* approaches are applicable when the scope of the dataset is finite and needs to be contrasted with generic browsers that are designed with the assumption that they can operate over arbitrary sources of Linked Data.

2.5.2.1 Faceted Browsers over Linked Data

Several Semantic Web interfaces have leveraged the power of faceted browsing, due to their relative ease of use, expressive power to construct complex queries, as well as preventing dead-end queries which yield no results. Since faceted browsing does not require an organisational hierarchy for the data beforehand, it has been a popular method for applying interfaces over graph data. Faceted browsers implementations over graph data vary in terms of how they provide mappings and configuration over data, how well they scale and various levels of functionalities they provide.

mSpace ([schraefel et al. \(2005\)](#)) is a faceted interface which arose from an attempt to generalise an implementation of an earlier interface - the CSAKTiveSpace [Shadbolt et al. \(2004\)](#). Typically used for collection of items about a particular topic, the mSpace is a faceted browser that allows the user to switch and select facets and arrange them in a hierarchy in which they wish to explore the information space (Figure 2.13). mSpace can be mapped on an RDF dataset through the use of the Facet Ontology - an abstraction over RDF data to provide mappings of RDF data to the mSpace data model ([Smith and mc schraefel \(2008\)](#)). Alternatively, the DataPicker interface allows easy installation of mSpace interface over a SPARQL repository of RDF data by picking a goal object (the items for which the mSpace is for) and select facets through connections in the graph ([Smith et al. \(2010\)](#)).

/facet is another faceted browser for RDF. */facet* allows a large number of facets to be simultaneously deployed on a faceted browser, by applying it to various collections of



Figure 2.13: The mSpace column faceted browser supporting backward highlighting.

items in a domain (Hildebrand et al. (2006)). For example, data about the Art domain can include collections about Artworks, Painters etc. and each of these collections have a number of facets. The tool allows the user to switch between collections to combine the filters from each collection. Thus, if the initial items were constrained by filtering and using some facets, the corresponding switching operation reflects this by showing only the corresponding items and facets.

To offer fast and responsive faceted browsing, all of the aforementioned tools require that the data is put in some database storage to facilitate fast querying over large datasets. Exhibit, on the other hand, provides faceted navigation interface over small datasets that can be easily deployed by a casual user knowledgeable only in HTML (Huynh et al. (2007b)). Exhibit intentionally hides the complexity of implementing a faceted browser by allowing users to only specify the using simple HTML constructs.

BrowseRDF is a faceted browser which allows faceted browsing without any need of an a priori configuration (Oren et al. (2006)). In addition to basic selection of facets, *BrowseRDF* allows *existential selection*, *join selection*, and *intersection* on facets as well as an inverse operation on these. Since it follows a non-configuration approach, the authors recognise that all properties in a particular collection of RDF data do not make suitable facets and propose a ranking mechanism to foreground potentially useful facets to the user. The ranking is done by viewing faceted browsing as constructing a decision tree, showing useful facets according to the *predicate balance*, *object cardinality* and *predicate frequency* after each faceted selection. This problem of identifying useful facets again gives rise to the problem of lenses - lenses can not only be utilised to show presentational aspects of the data, but also serve to denote other useful aspects; for example which groups of properties make good facets together for browsing.

2.5.2.2 Widget Library Approaches

Another common approach to generating interfaces over Linked Data is through the use of widget libraries that allow portions of data in RDF to be mapped to widgets. These widgets can then be assembled and added to interface implementations. In this respect

Exhibit and Dido can be viewed as widget libraries. [Hildebrand and van Ossenbruggen \(2009\)](#) allows autocompletion and faceted browsing to be used as off-the-self widget components. A similar approach is taken by Paggr which additionally allows data to be passed by widgets ([Nowack \(2009\)](#)).

2.5.3 Data-Browsers and Navigational Interfaces

The approaches described so far follow a common paradigm: they allow interfaces supporting interactions over data to be easily added on top of predetermined dataset. This allows the publishers of data-oriented interfaces total control over the experience and interactions of data. A distributed data-connected Web of Data, however, offers the possibility to access, integrate and use data on demand and not be limited by the user interface. Similar to how a Web browser allows access to any webpage on the Web, we need tools which allow us to access data - any data published on the Web. Facilitating access of data on demand, however, is significantly more complex. While the Web is a collection of documents where content and presentation are combined and routed to the browser, the Web of Data is raw information without any presentation. Thus, representation is left to the data consumer. Additionally, access and navigation on the Web of Documents is a relatively simple concept - access is one page at a time and navigation takes the user to another page. Links on Linked Data, on the other hand, are typed and this allows for richer navigation with access to one or several resources at a time.

Data Browsers are commonly used to facilitate access and navigation of Linked Data on demand. The simplest approach to navigating arbitrary graphs of data is to view any RDF resource as a web page. Browsers such as Disco²⁷, Marbles²⁸ and Zitgist²⁹ follow this approach. The browser renders one RDF resource per page, showing all the properties and property values for each RDF resource. If a value of a property links to another RDF resource, this is rendered as a hyperlink to that RDF resource. These browsers, however, do not offer much more utility over viewing the same information on a regular web page. For example, a Wikipedia article about Berlin is a much better experience than viewing the corresponding information in a generic browser's rendering of the Berlin DBPedia resource. While some of these types of browsers do provide lenses over data, they still do not add any additional value over Web documents. Haystack, a personal information management tool, also facilitates one resource at a time browsing, however multiple instances can be joined in collections for presentation purposes (e.g. multiple calendar events presented on a calendar) ([Quan and Karger \(2004\)](#)). In addition to presentations, Haystack also allows authoring of semantically described services that can invoke operations on data or resources that are in current view.

²⁷<http://www4.wiwiiss.fu-berlin.de/bizer/ng4j/disco/>

²⁸<http://www5.wiwiiss.fu-berlin.de/marbles/>

²⁹<http://dataviewer.zitgist.com/>

Pivoting or *set-oriented browsing* (sometimes referred to as *link-sliding*) is a technique of refocusing a view on a particular set of items by simultaneously navigating through a common property. It is an extension of the *one-to-one* browsing paradigm, which has been the prevalent browsing mode on the Web, to a *many-to-many* browsing mode.

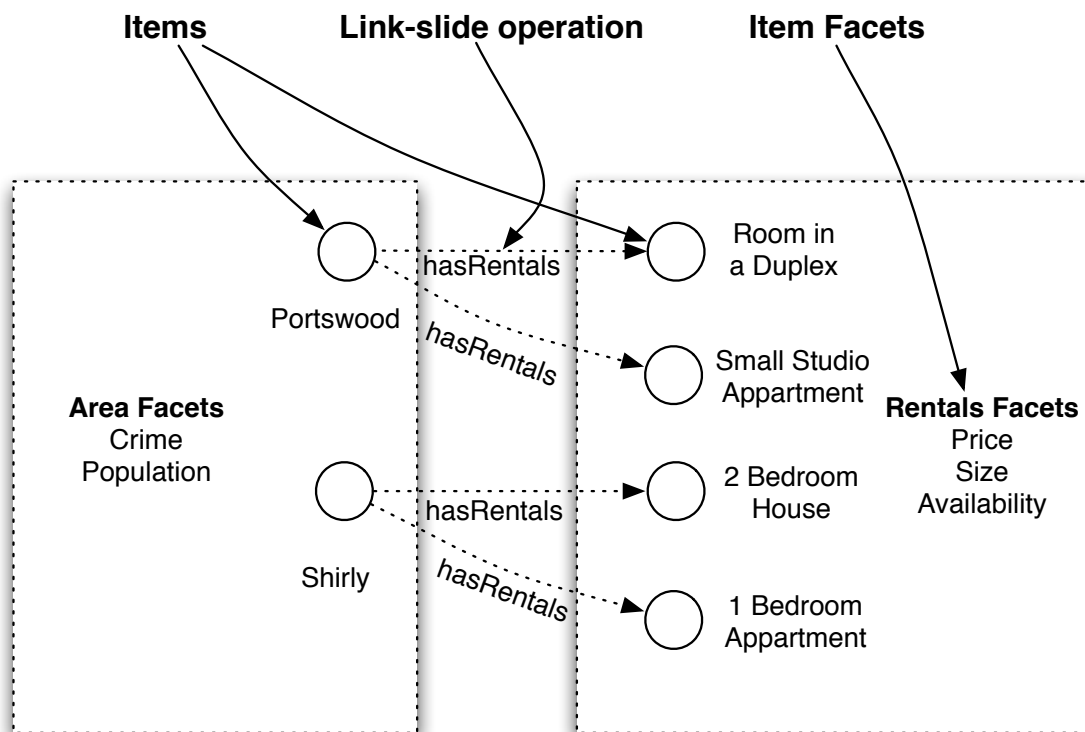


Figure 2.14: A pivoting example showing a pivot from a collection of places to rentals.

Pivoting is a more natural type of interaction with data rather than documents since the items of interaction are typically real-world concepts and properties have a meaning associated with them. The general concept of pivoting is shown through an example in Figure 2.14. In a browser supporting pivoting, users can access several resources at a time (in the example resource of type “*place*”). They then simultaneously get or *pivot* to all other available resources (in our case *rentals* items) through the common property (in this case the *hasRentals* property). Such interaction is currently unsupported on the document Web; to gather all of the required information a user would have to go through several pages and only one at a time in order to retrieve the required information. Pivoting over graph data reduces this to a single step by leveraging common properties in sets of resources.

Implementation of pivoting varies across browsers. The Tabulator (Figure 2.15) can be considered as an early example of pivoting (Berners-lee et al. (2006), Berners-Lee et al. (2007)). Users browse starting from a single resource following links to other resources. Tabulator then allows users to select a pattern by selecting fields in explored context and tabulate any results that are following the same pattern. Explorator uses pivoting

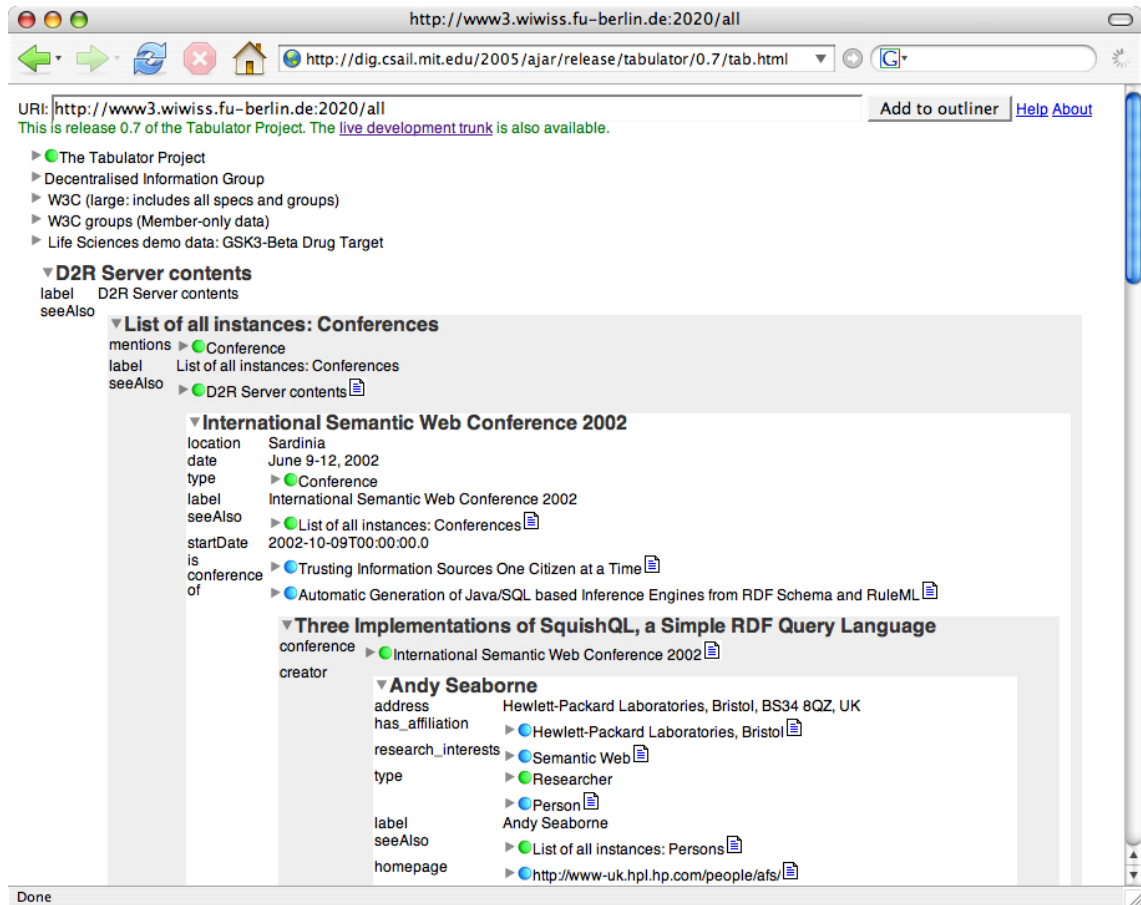


Figure 2.15: The Tabulator interface for exploring Linked Data. Each navigated resource is displayed as a nested concept of the resource from which the navigation originated.

as a metaphor for querying, where users select subjects, objects and predicates to create sets of things, subsequently combining them with unions and intersections operations (Araujo et al. (2009)). The Humboldt browser provides a list of items and faceted filters from which the user can choose to pivot or refocus (Kobilarov and Dickinson (2008)). Parallax (Figure 2.16) shows the current items, a list of facets and a list of connections showing the available properties to perform a pivoting operation (Huynh and Karger (2009b)). In VisiNav users can drag and drop properties and instances in order to pivot and filter through results (Harth (2009)). A common characteristic of these interfaces is the notion that pivoting never occurs in branching i.e. a user cannot pivot with two different properties from the current focus and keep the context of both trails of exploration. In Parallax, however, this is supported to some extent in the tabular view where generating a table allows this feature. gFacet also mitigates the problem of branching (Heim et al. (2010, 2008)). In gFacet, exploration starts from a collection of items. Users can get related lists of items through a selected property. The lists of items, generated through successive pivoting operations, are used as facets and spatially arranged in a graph visualisation.

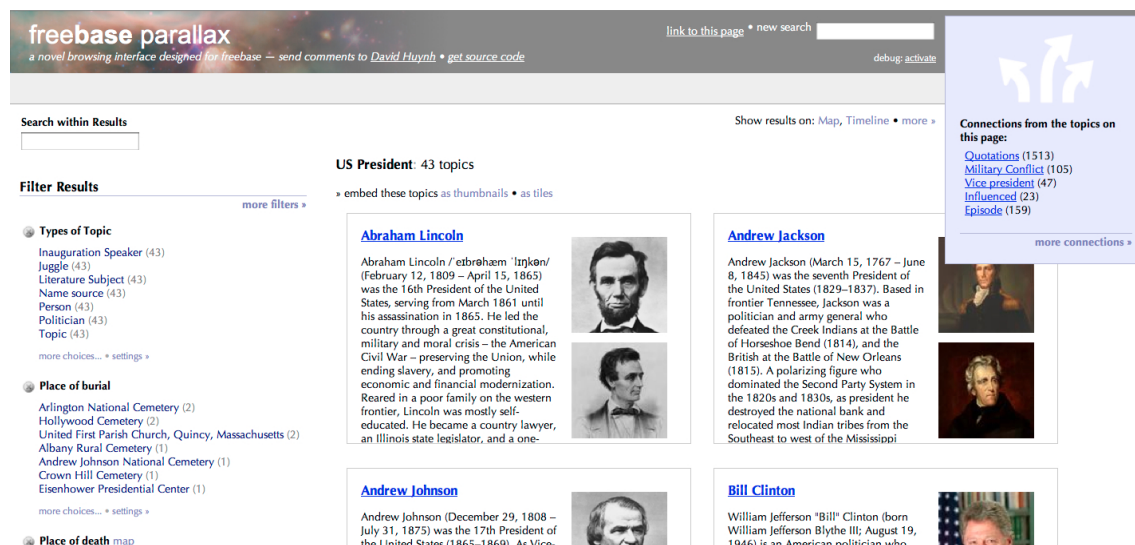


Figure 2.16: The Parallax browser. It allows set-oriented interaction with multiple resources. It supports faceted filtering of each collection (left) and a list of connection to which the collection can pivot (upper right).

2.5.4 Mashup Editors

Mashup editors and end user programming approaches have also been used as a template for building interfaces over arbitrary sources of Linked Data. This concept of visually building and querying for resources has been adopted by a number of interfaces over RDF data (Jarrar and Dikaiakos (2008); Morbidoni et al. (2007); Le-Phuoc et al. (2009)). Another approach to mashing data has been adopted by Potluck, which allows combining data from several Exhibits (Huynh et al. (2008)). Potluck uses drag and drop to combine resources from two Exhibits to create a single Exhibit of aggregated data. It also offers lightweight alignment tools to align potentially different data formats (e.g. editing telephone numbers - one that includes area codes and one that does not - across multiple cells simultaneously). Depending on their user target group, the complexity in all of the aforementioned approaches varies. In most cases, however, mashup editors are often too complex and time consuming for end users and do not offer enough flexibility to programmers.

2.5.5 Keyword Search and Natural-Language Processing

The browsers and data-centric tools presented so far are direct manipulation tools - they rely on users to interact the interface of data browsing and manipulate objects or input commands (textually or visually) to query for particular results. Another approach of data browsing is to rely on keyword search, a familiar and well-established way of finding information on the Web. Most of the semantic search engines (e.g. Sindice) use keyword search for entity retrieval of RDF resources (Tummarello et al. (2007)). Another text-based approach is providing Natural Language Interfaces (NLI). NLI use Natural

Language Processing (NLP) techniques to map complex queries required as free text to structured queries understood by computers. In a study performed by [Kaufmann and Bernstein \(2007\)](#), [Kaufmann and Bernstein \(2010\)](#), four variants of NL interfaces (NLI) were used to test the usability of natural language interfaces for querying structured data. The four interfaces included various levels of expressivity, starting from completely uncontrolled up to constrained language. The study showed that the full-sentence query option with feedback and iteration was significantly preferred to keywords, a menu-guided, and a graphical query language. The study, however, was done exclusively with well-formed fact-finding questions, i.e. questions that are not exploratory in nature. Thus, the performance of natural language tools versus direct manipulation tools for exploratory tasks is still an unexplored area.

2.6 Summary

This Chapter presented background work and a general overview of approaches to end user interfaces over Linked Data. It discussed the uses of structured data, and how interfaces can power novel ways of searching and querying for information over structured data. It also presented the core principles of Linked Data, and its aim to provide a global, distributed platform where data is published using common principles. The main points of this Chapter can be summarised as follows:

- Existing data browsing interfaces typically offer interactions over fixed datasets, thus fragmenting data sources by encapsulating them in tailor-made interfaces. Since Linked Data, in principle, allows data to be published using common standards and is linked to other resources before consumption, it potentially allows novel interfaces to be built, which would take the advantage of Linked Data over integrated data on demand to provide sense-making.
- The related work in Linked Data interfaces described two approaches of end-user interactions over data; the first attempts to generalise existing browsing interfaces, so that they can be easily configured over arbitrary datasets and thus deploy interfaces with ease, and second, generic data browsers that do not require a priori configuration over fixed data, but currently suffer from usability challenges because of their inherent generality. The Chapter also discussed some of the infrastructure and back-end limitations of deploying fully fledged, generic data-centric interfaces such as data browsers.

Chapter 3

Design Process for Data-centric Interactions over Linked Data

The original description of the Semantic Web envisioned applications that could wander through a linked world of thousands of different data resources, learning and discovering new sources of information in real time, and combining all the information to produce valuable answers to users. By enabling data-centric interactions, this thesis proposes placing the users directly in this loop; rather than having intelligent applications that gather information, users are empowered with tools that allow them to interrogate, explore and combine data sources on demand. Thus, such interfaces need to be generic, in the sense that they can access a source of Linked Data with no or minimal configuration, they need to be open-ended in design, meaning that users can serendipitously discover and add new data, and they need to be designed to handle the amounts of data which the Semantic Web envisions to be available on the Web.

Yet, as a class of applications we still know relatively little about design principles for data-centric interfaces designed for nontechnical end-users. Design challenges present themselves in several areas. The first challenge is defining good use-cases of data-centric interactions. In terms of higher-level information seeking processes, what sort of tasks are best supported by data-centric interactions? People use the Web for a wide array of information-related activities, from simple fact finding queries, to long-term information gathering ([Sellen et al. \(2002\)](#)). What sort of activities would benefit from having data-centric interfaces over Linked Data? Who are the potential users of such systems, and what biases and constraints do they bring to such interfaces? Upon answering such questions, we can begin diving into specific problems associated with tool design. For example, where does a user start exploring data? How is data browsing and exploration facilitated? How is data, represented internally as a graph, represented to users? How are individual resources represented? What additional tools would help users make sense and explore unknown datasets more efficiently?

With the increasing number of available data on the Web as Linked Data, a number of data-centric browsers have been proposed with end-users in mind. While these browsers share similar goals and rely on similar concepts - i.e. allowing end-users exploration, browsing and querying over arbitrary data sources - their implementations vary depending on how exploration is conducted, how data is represented, and how queries are executed. Therefore, there has been very little research on specific design considerations when building generic data browsers, leaving questions, such as which representations work best for which kinds of tasks, or comparing browsing techniques, largely unanswered.

To tackle the tasks of designing interfaces that offer data-centric interactions and data browsers in particular, this Chapter presents a design process framework. The design process is an attempt to formally approach the problem of designing data-centric browsers. It first attempts to characterise data-centric interactions by providing sufficiently realistic use case scenarios. Using these scenarios, the design process attempts to identify the attributes and challenges of designing data-centric browsers, by going through a requirements analysis exercise. By identifying attributes of data-centric browsers, it analyses and compares existing browsers on particular features and, when appropriate, suggests potential solutions from related areas. Combining this information, this Chapter describes an early prototyping effort of designing a generic data browser over Linked Data named GEORDi¹. Finally it synthesises the information gathered in the analysis to elicit a list of challenges associated with end-user access over Linked Data.

3.1 Design Process Goals

At its core, Linked Data is a method of publishing structured information on the Web about real-world entities. The links established between entities from local and distributed data sources reinforce an image of a Web of Data, where the basic unit of information is a resource of structured information that is addressable and accessible through a URI, in much the same way as documents on the Web. The image of a Web of Data intuitively suggests that one can replicate the notion of “browsing” data as an analogy to browsing documents on the Web. This premise has been the driving motivation in designing early data browsers over Linked Data. However, while access to a particular Web page is usually done in order to obtain information that has been intentionally packaged for consumption upon request, access of raw data is rarely a goal in itself - rather, it is the first step of solving a data-centric need, which includes the

¹A note on collaboration: The prototyping exercise described in this Chapter is partially a result of a collaboration among several researchers. The original designs of GEORDi were done in collaboration with Dr. Max Van Kleek. Implementation and coding of the prototype was done in collaboration among Dr. Max Van Kleek and Dr. Daniel A. Smith and myself. The observation and problems identified were the result of my own research.

ability to query, combine and analyse multiple data sources. For example, for typical end users, accessing structured, machine-readable information about London offers no better experience than accessing the Wikipedia page about London - in fact, without a presentational template the experience would be worse. On the other hand, accessing and comparing information for multiple cities, by making a custom chart visualisation might be achieved easier if structured data was available on the Web. This would save us time and effort of going through multiple web sites scraping the unstructured data to get an answer to the question. Thus, simply providing any access to structured information cannot be considered being a data-centric interaction - by that token, simply downloading a spreadsheet file on the Web might be considered a data-centric interaction.

In order to begin designing data-browsing interfaces we must clearly define what are high-level information-seeking activities that we want to support, and how Linked Data can be a contributor to having efficient data-centric interactions. Thus, the goal of this Chapter is to fashion a design process around which to investigate data-centric interactions over Linked Data. The described design process aims at gaining an understanding of the following:

- **Characterise the activity of browsing and sense-making over Linked Data.** What sort of information-seeking activities that require combining data do we want to support, and why does Linked Data provide a good platform for these types of activities?
- **Identify attributes of data-centric interfaces.** Existing instances of data-centric interfaces such as data browsers are inherently an entropic set of tools - they differ extensively in their intended audiences, their purpose and consequently the interactions they offer. How can we classify or compare data browsing interfaces? How are they different in design? What are the attributes along which these interfaces can be compared?
- **Identify problems and challenges.** By beginning to understand the high-level processes we want to support and provide a map of the attributes of data-centric browsers, and we can start to identify challenges to specific problems and potential solutions.

3.2 Method

For most design problems, the process exists in order to lend guidance to an inherently entropic set of tools and techniques. Such processes are first aimed at getting the right design i.e. selecting the best idea in a pool of many ideas, and then at getting the design right i.e. executing the chosen idea (Tohidi et al. (2006)). However, the initial challenge we are confronted with when approaching this particular design space is not

only generating ideas and selecting (and possibly executing) the best one, but identifying specific areas where browsing/exploring/interacting with Linked Data represents a challenge that needs to be addressed.

In order to create a design process to uncover as many as these areas as possible we used a method inspired by the scenario-based design ([Rosson and Carroll \(2003\)](#)) ([Figure 3.1](#)). The design process method described throughout this Chapter includes five steps:

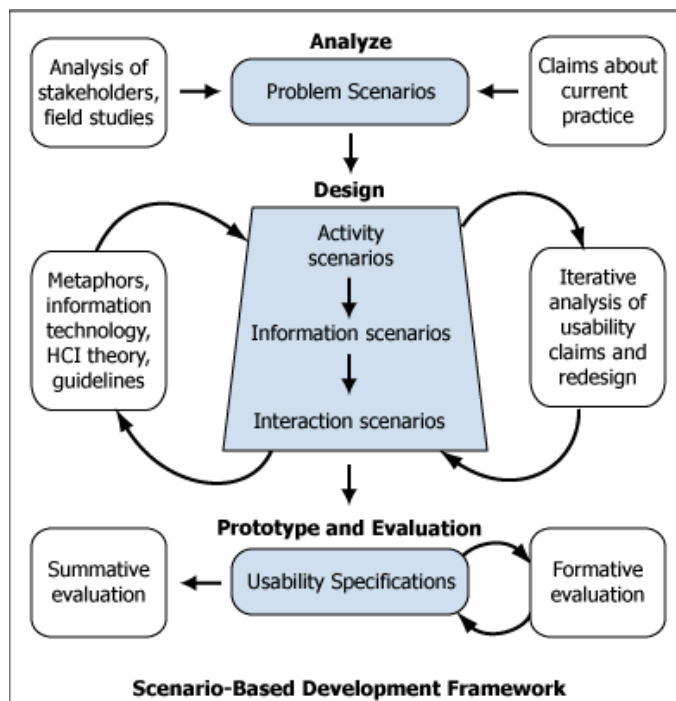


Figure 3.1: The Scenario-based design process proposed by [Rosson and Carroll \(2003\)](#).

- Propose personas** The design process begins by portraying personas which describe situations where data-centric interactions are required to solve an information-seeking task. In addition to portraying two data-centric interactions we also include descriptions of the types of stakeholders in each one, in order to suggest users with different backgrounds, skills and biases.
- Break down the process of data-centric interactions into specific activities** The next step in the design process is to break down the scenarios described in the personas into specific activities/requirements. A functional analysis of the requirements is carried out by looking at what tools currently support the accomplishment of the task outlined in the scenario/use case and identifying potential gaps. This analysis is described as a walkthrough - it lists the activities that the user needs to take in order to accomplish the task by using information sources and tools available on the document Web. We then make a similar iteration over the same scenario, based on the assumption that now we have a case where raw

information sources of Linked Data are available that can hypothetically be used to solve the same task. During the second iteration we try to identify problems in accessing Linked Data and try to identify existing research, HCI theory, and/or tools over interfaces over structured data which can potentially inform possible approaches to solving the identified problems.

- **Analyse existing interfaces and approaches offering data-centric interactions over Linked Data** Once specific activities/functions are identified in accessing Linked Data, we can use these to analyse the solutions presented in the existing set of tools that offer end-user access over Linked Data. For each of the listed challenges, we examine the corresponding component or interaction of the tool, note similarities or differences in their approaches and identify challenges for interaction and application design. When needed to illustrate how each tool approaches an identified challenge, we resort to describing the component through the hypothetical Linked Data sources provided earlier. Unfortunately, further testing of these tools on live data was not an option, since many of these tools are either inactive or publicly unavailable.
- **Prototyping, evaluation and reflection** Based on the requirements gained through the analysis, we engage in a prototyping exercise by designing a generic data browser named GEORDi. The purpose of this prototyping exercise is to iterate over the initial set of identified challenges and use deployment over live Linked Data in order to either refine these challenges or identify new ones. We outline specific design goals for GEORDi and include initial observations by testing execution of data-driven tasks over several different datasets.
- **Identify Challenges** Based on the walkthrough, an analysis of existing systems, and our own experience in prototyping, we can begin by proposing an initial set of challenges for building usable generic data browsers. These challenges can either be functions or affordances that current tools do not provide but are deemed necessary to complete a task or they might be design issues with functionalities provided in existing browsers. Challenges need to be directly associated to specific tasks described in the scenarios, which in turn can be attributed to a specific function in a data-centric browser.

3.3 Personas for Data-centric Interactions

The hypothesis for approaching the problem of creating data-centric browsers is that we will generate the most useful results if we focus this interrogation through a sufficiently challenging and realistic use case. To choose appropriate personas requires several criteria: (1) the persona must describe a situation where a task cannot be completed without

consulting multiple dataset, (2) personas can be differentiated based on the setting, context and current tools used to integrate data from multiple sources. Thus, we choose two personas inspired from both documented problems in end-user data interactions and real-life experience. Our personas are also based on two types of stakeholders. The first type of stakeholder is a data-journalist, a typical knowledge worker, familiar with end-user data manipulation tools such as spreadsheets. This type of user, however, rarely possesses advanced technical skills, such as programming and database administration. The second type of stakeholder is a graduate student who is largely unfamiliar with end-user data manipulation software. Rather, all searches and combining of information are done manually using the Web and possibly with the use of note-taking tools in order to combine and log information. To keep the personas sufficiently realistic we base our examples on real life situations. For the data-journalist stakeholder, the example scenarios are derived from several examples in the Data Journalism Handbook (Gray et al. (2012)). For our graduate student, the chosen scenario was based on the real life experience of graduate students when making travel arrangements to attend conferences abroad.

Persona 1

Chris, is a 32 year old data journalist working for the Economist. Chris is currently working on an investigative piece on the increasing rise and development of third-world countries. Over the past 10 years multiple articles have suggested that many developing countries have made great strides towards becoming developed countries themselves, and in fact so much that it becomes increasingly meaningless to classify all these countries as simply the “developing. Indeed, enormous differences in development levels exist among these countries. Chris’s current piece attempts to create a comprehensive study supported by various data sources examining this phenomenon from different angles using different sets of data. For example, he would like to include and examine how this phenomenon is distributed along different regions in the world by examining GDP growth, poverty levels, trade etc. In addition to looking at standard development indicators, he wants to bring in additional factors to his analysis; for example information about how much of this development has been tied to investments in infrastructure projects such as transportation systems, or how is governance and corruption in particular correlated to development indicators. In addition to social and economic indicators, he would also like to examine cultural indicators that suggest that developing countries are catching up. For example, he would like to see the number of developing countries that have hosted major world events such as the Olympic Games and the World Cup, which have historically been hosted by more developed countries. Chris is a computer-literate professional that is knowledgeable in manipulating spreadsheets of data and visualising them

by using charts and graphs. However, for many of the unstructured sources he wants to integrate, he would need the services of a programmer.

Persona 2

Anna, is a 25 year old UK graduate student working on her PhD in Neuroscience. Anna just published her first paper at an international conference in Lyon, France. In order to attend the conference, except for registering, she also needs travel arrangements, which include booking a hotel and a flight to Lyon. Being a PhD student, Anna knows that her annual travel budget is limited, so she needs to find suitable accommodation and travel arrangements at a reasonable price. In addition to the price limitation, she also has other requirements; she wants the hotel to be relatively close to the conference centre or at least close to a local transportation station with links going to the conference centre. Furthermore, she would like to be able to visit the old part of city, so she wants the hotel to be also accessible to that part of the town. To find good prices, she knows that she can get deals if she were to combine her flight with her hotel booking on a variety of travel sites. Additionally, the conference has listed several hotels on their site that provide discount prices for conference attendees. Anna has previously used and usually bookmarks pages of information when her tasks require gathering information from multiple websites.

3.3.1 Walkthrough

In the following section we provide a walkthrough and examine the challenges in accomplishing each of the tasks with available resources on the Web today.

Walkthrough 1

Chris's task at hand requires him to do several things. First, he needs to find as much information he can about the various attributes or data he wants to include in his analysis about developing countries and then he wants to be able to iteratively explore the data as a whole i.e. be able to filter and compare different countries by various criteria in order to find interesting patterns to report. Chris imagines that his task would involve generating a customised spreadsheet that aggregates data about the various things he wants to examine. For example, he would imagine that his end result would be a spreadsheet that starts with a column of all the developing countries and have all the subsequent columns contain data related to the countries. The additional columns, for example, would list the countries GDP per capita, development index, inequality index, number of infrastructure investments, number of hosted Olympic games or World Cups and other data he might deem important to his analysis.

To complete his first task he starts by searching for development data about countries from websites known to him such as the World Bank², the IMF³ and UN statistics site⁴. These sites offer a number of ways to explore what sort of data each dataset contains. For example, the World Bank offers a selection of indicators, along with a written explanation of the indicator.

Once these datasets are examined, Chris decides that the best sources of data are the IMF and World Bank datasets. He needs to combine both for different purposes; however he faces challenges. For example, while the World Bank data contains good data about various development indicators, these refer to all of the world countries and do not provide any groups. The IMF dataset, on the other hand, provides groups of countries as filters, including developing countries. Chris wants to use the IMF list of developed countries to filter the countries in the World Bank dataset and get the indicator data only for the corresponding countries. One sign of relief for Chris is the fact that these sites provide data in structured format, that is, as spreadsheets. However, even with having the data in a structured format he faces challenges. He notices that certain countries are named differently in the respective datasets. For example, the name of the country of China in the World Bank dataset simply stands as “China”, while the IMF dataset names China as the “People’s Republic of China”. Thus, once he copies and pastes the data in a single spreadsheet, he has the task of reconciling these differences. Every additional dataset he wants to add requires the tedious task of checking the names of the countries in the dataset with the names of the countries in the aggregate spreadsheet. This problem includes adding data that is not directly relevant to the analysis, but rather used for filtering or grouping purposes. For example, he sorts the data by GDP per capita; however he wants to group the countries by region in order to examine which are the highest developed ones by this criterion. Since none of the current datasets include a region, he needs to enter this information manually or find another source of information, organize it if it is unstructured, and reconcile the data.

After analysing some preliminary data, Chris wants to include additional data that showcases how these development levels are showcased in specific visible examples. For example, he needs to find data about the total investments in infrastructure projects, such as transportation and energy systems. He finds data about both requirements, but he notices that there is no data about all countries, and rather per regions within countries. For example, transportation systems such as subways are broken down by city data. Thus, Chris must go through the arduous task of finding the country for each of these cities and then combine the data to get the full information. The same problem occurs when trying to find how much of the cultural events (e.g. the Olympic Games) are now hosted by developing countries. He finds unstructured data on the Wikipedia⁵

²<http://data.worldbank.org/>

³<http://www.imf.org/external/data.htm>

⁴<http://unstats.un.org/unsd/default.htm>

⁵<http://wikipedia.org>

for Olympic Games, but the information on the hosts is only by city. Once again, in addition to structuring unstructured data, he needs to find the corresponding country for each city and combine them to find the number of sporting events that each country has hosted. Once all the data Chris needs is in place, he can try to further filter or visualise the aggregated set in order to find interesting patterns for his report.

Walkthrough 2

Anna first opens the conference web site and navigates to the “Attendees” section. She notices that the conference organisers have listed twelve hotels that offer lower rates for conference attendees. She then opens another tab on her browser and tries to make travel arrangements for flying from London to Lyon through popular sites such as Expedia⁶ and Kayak⁷. Unfortunately, both websites offer a long list of over 150 offers of flights and hotels for the specified dates, and Anna is left to herself to figure out if any of these include the twelve hotels listed on the conference website. She also realises that she needs to find if any of the offers are in fact cheaper with the discount offered by the conference. Thus, Anna is left with the task of manually looking up the price for each hotel individually. Some websites offer just a search by inputting travel dates and do not allow searching by hotel; in those cases Anna needs to find the hotel in the list of search results. In order to keep the information organised, she opens up a notepad and copies the 12 hotels from the conference website. Going through the data she can see that there are some hotels which in fact cheaper than the ones listed on the conference website besides the discount, but not by much. To make a decision she tries to look up reviews about these hotels on TripAdvisor⁸. Again, she needs to go through the task of looking up one hotel at a time, recording the ratings in her notepad for each hotel. Finally, she wants to include information about the geographical location of these hotels in order to see if any of them are far away from the conference centre and, if appropriate, whether transport links are available. Most of the travel websites allow hotels to be viewed on a map; however once again Anna cannot filter the hotels she is already considering in her notepad; rather she can only display all the hotels by going back to the search results. Given the large number of results, the map is of a very little use to her. So she decides to use Google Maps⁹ in order to create a custom map and input the hotel addresses one by one. However, now she faces problems with filtering the ones which have access to good transportation links. While Google map provides information about bus and tram stops, it does not include information about transport lines. Anna finds a dedicated website about the city transportation that allows her to input addresses, and the website lists the available transport routes along with transport times. Thus, she is again forced to look up and input the address for each

⁶<http://expedia.co.uk/>

⁷<http://www.kayak.com/>

⁸<http://tripadvisor.com>

⁹<http://www.google.com/>

hotel individually. Finally, after multiple iterations taking a couple of hours, she finally decides on a particular hotel.

3.3.2 Task analysis

Chriss and Anna's tasks, while specific to their needs, represent typical search engagements expressed as information retrieval and sense-making tasks (summarised in [Wilson et al. \(2009\)](#)): (1) source discovery - initial review of kinds of needed information, seeking out available sources (2) triage - from the available sources, assess each one rapidly to see if it is worth further probing; (3) once a data set is determined to be useful, interrogate it further in order to produce a result; (4) Integrate multiple data sources and organise information in a way that is required to complete a task; (5) represent the data in a way that allows insight or completion of a task. These tasks are both very similar to the sense-making model proposed by [Pirolli and Card \(2005\)](#). The sense-making model proposed by [Pirolli and Card \(2005\)](#) includes two major loops - a foraging loop where users iteratively search, find, filter and query for information, and a sense-making loop where users engage in an iteration of schematising or organising this information, stating hypothesis and representing data that provides insight. While the sense-making model proposed by [Pirolli and Card \(2005\)](#) was developed by observing analysts and specialised knowledge workers in sense-making tasks, we can observe that both tasks described in the walkthroughs, while specific in the type of task and accessed information, exhibit general similarities with this model. In the following section we break down and examine the various functions that Chris and Anna are engaged in, while performing their task.

Source Discovery. Chris's and Anna's tasks both start by finding relevant information sources, in their case websites, where they can search for information pertaining to their tasks. In Chris's case, he starts by searching for information on a number of known data portals and in Anna's case, web sites offering travel information services. Such tasks present minor challenges since available sources of information provided on a website are easily searchable through web search engines.

Triage. For the second stage of their quest, both Chris and Anna need to interrogate if the information in the information sources under consideration is of any use in completing their tasks. We can notice that this is done per information source - for example Chris can filter through the various attributes on the World Bank data website to see the various data he can find about countries. Triage of information is usually well supported on websites powered by structured information since all the data provided is known at the time of development and thus different ways of interacting with the data can be anticipated and accounted through development of the interface. As mentioned in Chapter 2, many websites utilise powerful browsing techniques such as filtering, keyword search and visualisation to provide customised search of data. In our examples, the World Bank data website allows users to explore, chart and filter for the datasets

provided by the website. Similarly, travel sites, such as Expedia, aim at providing high quality of query functions to users in order to efficiently find the best travel offers.

Data integration, organisation and schematisation. Going through both walkthroughs, one can easily notice that the most challenging part of Chriss and Annas tasks are integrating multiple sources of information that are located on distributed websites. As we can notice from the given examples, none of the websites that Chris and Anna access neither provide all the required data to solve their tasks, nor do they allow adding external data sources. Since these sites provide only custom views and interactions of their data, both Chris and Anna are forced to manually try to perform the task of data integration. For example, as described in the tasks, both of them need external data sources or services over external data sources for filtering purposes. For example, Chris needs to find the groups of (developing) countries provided in the IMF dataset and filter the data he can use in the World Bank dataset. Similarly, Anna needs the hotels provided on the conference website in order to filter out the ones that are provided in the search results on Expedia. We notice a similar problem when they try to merge data sources. For example, Chris's task of merging data from different countries is made difficult by the fact that the countries are not named the same across datasets. We also see that merging of data is difficult in cases where information is not provided on equivalent information resources. For example, when Chris tries to find data about Olympic Games hosted by countries, he finds only information about the city host. Therefore he needs to manually find each country for the city in the dataset so he can effectively merge the data to his existing spreadsheet centred on countries. Moreover, we can notice that these sort of data integration tasks are often time consuming because they often require working with multiple data resources at the same time. For example, Chris needs to aggregate data around a set of countries (i.e. developing countries) instead of one particular country. Anna needs to work with multiple hotels at a time. In order to do data integration, both Chris and Anna resort to improvisations and customised organisation of their information. In Chris's case he needs to combine and organise his information in a spreadsheet, while Anna uses a note-taking tool to record various information she encounters from the websites she uses.

Legibility and Data Transformation. Besides integrating various data sources, other challenges of using distributed data sources are: customised representation or legibility used for the purposes of further analysis, and additional data transformations. For example, in our second example, Anna tries to create a geographic representation of the data about hotels by inputting them in a custom Google Maps map. This again, requires her to input one data resource at a time. On the other hand, Chris faces data transformation challenges in combining his data. For example, rather than the actual information which lists the Olympic Games hosted by city, he only needs the sum of each one grouped by country. Thus, he is forced to do this sort of manual data transformation in order to get the corresponding count of Olympic Games hosted by each country.

3.3.2.1 Stakeholders

As a final analysis of our tasks we also need to examine and note the differences in the stakeholders described in these personas. As we have noticed, both tasks exhibit similar patterns. However, their approaches differ in noticeable ways. Chris is a knowledge worker and his approach to solving his task is very much influenced by his skills. While Chris does not have advanced programming skills to either query a database or write a script to structure data, he is comfortable with end user data tools such as spreadsheets and can thus handle raw information sources, if these are provided in a suitable and understandable format. Unlike Chris, however, Anna's use of information is strictly confined to the access of available tools on the Web. Her view and ability to query and visualise data is strictly confined to the availability of an end user application that is domain specific. This is also obvious from the fact that Anna uses no data model or more formal representations in organising her information. For example, most of the information she records is scrappy note-taking i.e. it does not exhibit any strong structure as, for example, spreadsheets do. As we can see, she collects information about hotel addresses and then uses an end user tool, in this case Google Maps, to display the hotels geographically.

Another important note to make is the differences in the structure of information Chris and Anna want to access. From the personas and walkthroughs we notice that Chris is purposefully looking at more raw information sources. While the World Bank dataset allows plotting charts directly from their website, Chris is more interested in making his own analysis and visualisations in order to find data patterns. Anna's exploration of the data is more task-oriented - combining data is done for the purpose of accomplishing a task and not engaging in an in-depth exploratory search or data analysis. Thus, for Anna, who represents a typical "Web user", even if more raw structured information for her task were to be available, it is unlikely that she would interact with raw data representations and manual visualisations over data.

3.3.3 Linked Data Scenario

In the subsequent steps of our design process, we concentrate our efforts on the first scenario. The reason why we choose to tackle the first scenario and the first type of stakeholders was based on two factors: (1) most of the related work in this area included generic browsing interfaces that used generic representations of data to provide access to Linked Data on demand and (2) based on the notion that we were also building a generic tool that could access Linked Data on demand, some sort of generic data representation design was intuitively suggested; given the substantial validation of spreadsheet software as the most used end-user generic data tool, our initial scenario seemed a more natural fit. Indeed our initial design brainstorming sessions considered that a generic data browser

offering the easy use of spreadsheets might naturally extend to users who were previously unfamiliar with spreadsheet software. Thus, many of the observations on designing our first type of stakeholder informed us how much complexity could be handled by a person with even less data-related skills.

In this section we examine a hypothetical scenario in which our first persona, Chris, instead of iterating and collecting data from sources on the document Web, is able to iterate over Linked Data sources that can potentially help him solve his task more efficiently i.e. help him face the myriad of data integration problems in solving his task by combining data about developing countries. Our scenario is used in order to identify the gaps or areas where Chris would have difficulties in accessing and interacting with Linked Data. Given a sufficiently realistic example of available and relevant Linked Data published on the Web, we raise the question of how would the tasks Chris is engaged with be performed over Linked Data on the Web. For example, how can he find datasets of Linked Data, how can he triage and interrogate to see if any of these have potential relevant sources, what sort of data interactions would he be required to do in order to find and combine all the data he needs for developed countries and what are the inherent challenges in accessing data over documents.

In order to select our scenario, we first examine the current set of Linked Data publishing practices and provide an example of Linked Data data space centred on these data publishing practices. Given these data publishing principles we examine the anatomy of a generic data browser i.e. around what principles can a generic data browser be designed that allow access to Linked Data sources on demand based on current publishing practices. Finally, we provide an overview of the challenges of executing the task over Linked Data, by examining the gaps between the task described in the walkthrough and a hypothetical scenario using a generic data browser.

3.3.3.1 Linked Data Publishing Practices

The principles of publishing Linked Data are outlined in [Berners-Lee \(2006\)](#). As noted in Chapter 2, Linked Data sources should be available by dereferencing a URI i.e. every resource named with a URI should be accessible through dereferencing that URI. The third principle requires returning useful information upon request, and while there has been a debate if the RDF standard should be mandatory upon a recommendation for modelling data sources (see [Does Linked Data need RDF?](#)¹⁰), most of the Linked Data available on the Web today is published using the RDF standard. Apart from dereferencing URIs, a common way to provide access to Linked Data is allowing access through a SPARQL endpoints, which allow database-like queries to be executed over RDF by remote clients ([Prudhommeaux and Seaborne \(2008\)](#)). While the rest of technology standards which are part of the Semantic Web stack are compatible to use when

¹⁰<http://cloudofdata.com/2009/07/does-linked-data-need-rdf/>

publishing Linked Data, they are not part of the principles behind publishing Linked Data.

As a global distributed space of data published on the Web, most of the data published as Linked Data is clustered around Linked Datasets - data sources published and/or maintained by a single publishing entity. Thus, rather than looking at one homogeneous global data repository, Linked Data, for the most part, represents a global dataset of repositories linked between each other on the resource definition layer. This is an important observation, since searching for data sources can be based on datasets as well as searching Linked Data on the Web as a whole. To describe datasets, VoID, a machine readable ontology has been proposed to publish structured data describing a Linked Dataset published on the Web ([Alexander et al. \(2009\)](#)).

Another important thing to note is the use of data vocabularies on the Web of Linked Data. Whenever Linked Data is published, a data vocabulary is defined to describe the data. These include naming of properties and classes of data, which are the staple of the structure that this data will use. When publishing Linked Data, reuse of existing vocabularies is encouraged when possible ([Heath and Bizer \(2011\)](#)). Statistics over Linked Data, however, suggest that the reuse of vocabularies is mostly concentrated around a few vocabularies (such as the FOAF¹¹, Dublin Core¹², SKOS¹³, RDFS¹⁴ etc.), which tend to define properties, such as attributes for human readable labels, identification of classes, specifying hierarchies etc ([Bizer et al. \(2011\)](#)).

Finally, we should note that publishing principles are evolving as Linked Data technology matures and many problems and errors in publishing do persist ([Hogan et al. \(2010\)](#)). In general, we can rely on these current publishing practices as a guide for this purpose of designing tools over arbitrary sources of Linked Data.

3.3.3.2 Anatomy of a Linked Data Generic Browser

Generic interfaces offering raw data access over Linked Data, should theoretically provide access to Linked Data on demand without any or with minimal effort of configuration by the end user. Given the current publishing practices of Linked Data stated earlier, a generic browser can provide interaction over Linked Data based on a minimal set of certainties about the data:

- **The common triple-based data model of RDF** A generic data browser can currently only base its interaction upon the “knowledge” that the data is modelled using the RDF data model. Access from a data browser can thus be done at the

¹¹<http://www.foaf-project.org/>

¹²<http://dublincore.org/>

¹³<http://www.w3.org/2004/02/skos/>

¹⁴<http://www.w3.org/TR/rdf-schema/>

level of RDF resources - either accessing a single or multiple RDF resources at a time. Additionally, since the data model is known (i.e. it is a graph model), navigation is permitted to access additional resources - i.e. accessing a link in one or a common link (equivalent property) in multiple resources will return the corresponding link resource(s). An example of an early generic data browser is depicted in Figure 3.2, displaying an RDF resource of a person.

- **Use of heuristics over known vocabularies** In order to display data to a user, a generic browser needs to choose how the data will be rendered in the browser. The most obvious solution is to use its “knowledge” of the triple-based data model to provide a list of property - object tuples for each resource currently under inspection in the browser. However, such data rendering will yield data representation in machine readable format. For a human readable representation a browser needs to access the human readable labels in the data (assuming the data is published with the best practices and has human readable labels - for statistics on the use of human readable labels see [Ell et al. \(2011\)](#)). This means that for every property and object value a human readable label must be accessed. Given commonly used vocabularies, a Linked Data browser might search for human readable information. For example, `rdfs:label` and a `foaf:name` are commonly used properties to find the human readable labels for RDF resources.

3.3.3.3 Available Linked Data

Based on the description of the tasks in the case of our first persona, let us hypothetically assume that data relevant for the task is available on the Web as Linked Data. In order to better illustrate the later described challenges, we provide a snapshot of example sources of Linked Data relevant for the task described in our first persona. Figure 3.3 depicts an ontology representation of a subset Linked Data with data relevant for Chris’s task. In the Figure, we use circles that represent classes, and the inside rectangles denote literal properties on instances of these classes. For example, “Countries” have literal properties such as “Label” and “GPD per Capita” and links to resources of other classes such as “Politicians” and “Cities”. Instances of these classes have themselves literal properties and links to other resources, for example “Cities” have “Name” and “Population” properties and links to “Subways” and “Olympic Games”. The relationships between resources are displayed as arcs, which denote that instances between two classes can be linked through a particular property. Finally, as the Figure shows, various data might reside in different datasets, although there also might be a case where the entire data resides or is republished in one dataset.



Figure 3.2: The Disco Generic Data Browser.

3.3.3.4 Challenges in Interacting over Linked Data

Given that relevant Linked Data sources are available on the Web, let us examine the challenges Chris would encounter using an end user data-centric tool, such as generic data browser to access the data. A generic data browser would need to facilitate all the processes Chris needs to go through: finding potentially relevant data sources, interrogating them, exploring and integrating relevant data sources, and using the data in a way that facilitates analysis and answers questions..

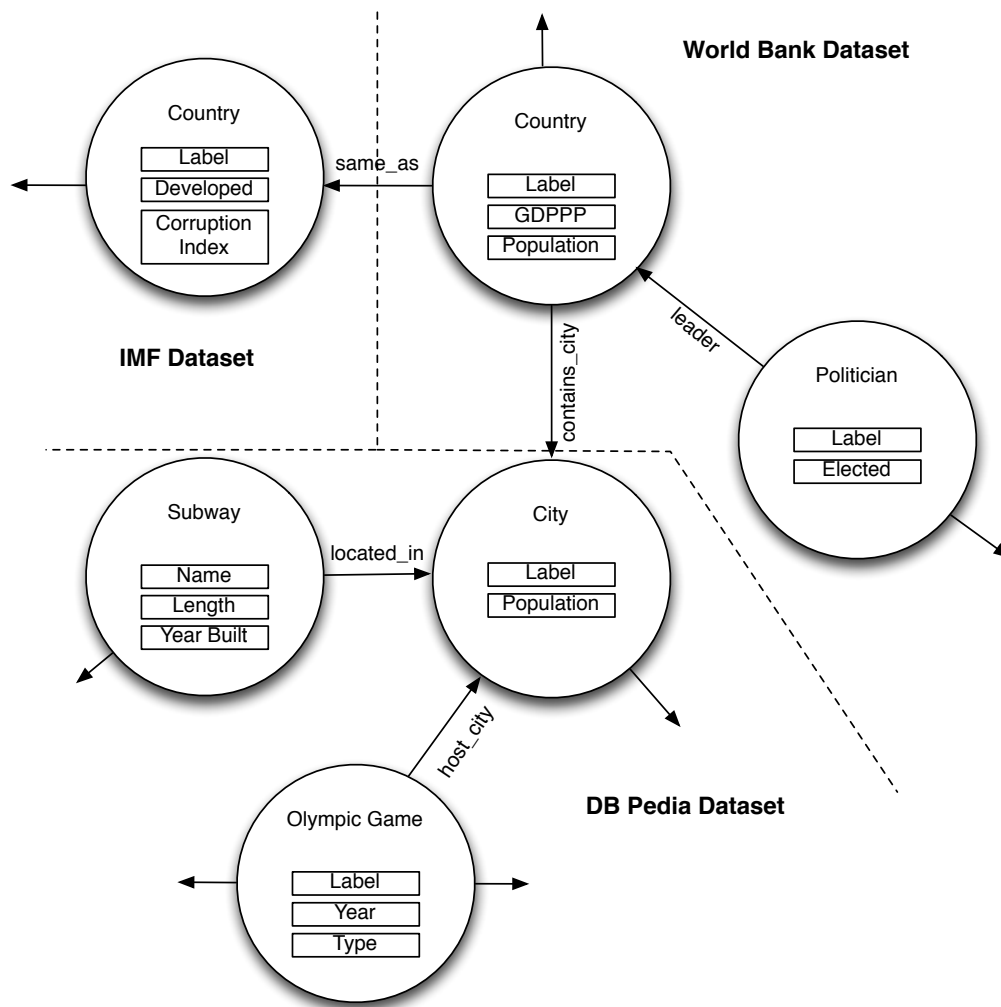


Figure 3.3: A conceptual ontology for data about UK Members of Parliament.

Dataset discovery and exploration starting point. The initial challenge for Chris would be to find suitable Linked Datasets that might potentially have relevant information. Even finding one potentially relevant dataset might be useful, since links to other data sources might provide him with a way of accessing all the other data he needs. For example, by finding the countries in the World Bank Linked Dataset he can follow links to other data he requires.

We notice that on the document Web, Chris can search for repositories of data which are accessible through a Web interface. Sites offering data sources, such as the World Bank or Google Public Data¹⁵, offer catalogue-like features for finding relevant datasets. In contrast, very few services exist that offer similar services for Linked Datasets. Descriptions about Linked Datasets, are usually provided in Linked Data format themselves,

¹⁵<http://www.google.com/publicdata/directory>

using vocabularies such as VoID ([Alexander et al. \(2009\)](#)). Aggregations of VoID descriptions or VoID stores^{16 17} exist to query over such data. However, no human accessible interface allows a user to browse or explore aggregates of VoID descriptions. These services can be easily implemented if a standardised vocabulary is assumed. A more important problem than finding potentially relevant datasets is how can end users, such as Chris, start exploring an identified dataset, assuming one can be found in a catalogue. Should a generic browser require him to input a URI, similar to inputting URL in a Web browser? How would Chris start exploring the content of an unfamiliar dataset? What should the access to a dataset provide to him in order to begin interrogating a dataset?

Data Exploration, Navigation and Representation. As we've noted, from a purely technology viewpoint, the purpose of a generic data browser is to provide the users with an access to Linked Data resources and the ability to browse or navigate through related resources via semantically typed links. However, how do such browsers enable Chris to perform his tasks? For example, let us assume that Chris is able to retrieve the country resources from the World Bank Dataset. How does he browse through all the data he requires - country data from the IMF dataset, data about infrastructure projects, to Olympic Games? As we can note from the example in Figure 3.3, in some cases Chris would have to navigate several steps from his initial starting data resources in order to get relevant data. How, in the absence of knowledge of a dataset, can he effectively navigate, query and filter data from a complex and densely connected graph? Finally, how would browsing in a generic browser be best represented to (1) make Chris understand complex graphs of data in a way that would allow him to formulate a query, (2) how would navigation steps and exploration of data through navigation be represented in a browser in a way that is compatible with his task?

Tools for representation and analysis. Finally, assuming that a generic browser is designed to support sense-making tasks, how can data access be combined with easy-to-use tools for representing and analysing data? Can data transformations and visualisations which are a key enabler of sense-making be easily made available over graph data accessed from a generic browser ([Russell et al. \(2008\)](#)). For example, spreadsheets allow users to create and organise data; however they also provide end users with tools for data transformation and visualisation.

3.4 Analysis of Interfaces Supporting Data-centric Interactions over Linked Data

In this section we analyse existing instances of data browsers and examine the different ways they provide data-centric interactions over Linked Data. Our selection includes

¹⁶<http://kwijibo.talis.com/voID/Describer>

¹⁷<http://void.rkbexplorer.com/>

ten different data browsers, which we select based on the following criteria:

- The browser needs to be designed for end users with no or minimal knowledge of data-related technologies.
- It supports navigation, browsing and interaction over graph data on demand.
- It is specifically designed to explore, browse and query data to support information seeking needs.
- It allows interactions over sets of resources instead of individual resources. We chose this criteria since data interactions usually require querying, filtering and operations that allow interaction over several data resources. By resources, in this context, we mean the smallest unit of information presented in the browser. Since all browsers use a single RDF resource as the smallest unit of information the term resource can be considered synonymous with an RDF resource. Browsers such as Disco¹⁸, Marbles¹⁹ and Zitgist²⁰ are not considered, because they support simple RDF browsing with individual RDF resources displayed as Web documents and thus do not provide any exploration or query capabilities.

Throughout this section we use the example Linked Data depicted in Figure 3.3 to illustrate concepts and interactions in various browsers. Based on the challenges identified in the previous section we examine the browsers in the following areas:

- **Exploration Starting Point** How do users initiate an exploration task in a data browser? Data browsers need to provide a starting point where the user can enter an initial input or query.
- **Navigation and Browsing** How do users navigate and browse through data in the browser? Are there different aspects of browsing and how are they supported? What affordances does each of these potentially provide? Do browsers rely only on graph navigation to explore relations between remote resources or are other tools provided? How is the context between resources accessed in navigation shown between several browsing steps?
- **Data representation** How is data represented in the browser? Data representation is examined at two levels: first how the data is presented of the lowest data granularity offered in the browser (e.g. a RDF resource) and second, how is navigation represented and visualised between resources of data.
- **Query and Filtering** How is data querying and filtering supported in the system?

¹⁸<http://www4.wiwiiss.fu-berlin.de/bizer/ng4j/disco/>

¹⁹<http://marbles.sourceforge.net/>

²⁰<http://dataviewer.zitgist.com/>

- **Tools for representation and analysis** What additional utilities does the tool provide to add different data representations apart from the generic one, and what tools are offered for further analysis?

3.4.1 Exploration Starting Point

Most browsers have a notion of a starting point of exploration, an initial input that results in some data after which the user can use the navigational and filtering features of the user interfaces. As we noted, while Linked Data is a wholly global distributed space of data on the Web, publishing practices suggest that data will be clustered in datasets where datasets are repositories of data published by a single identifiable publisher. Thus, a data browser could take the publication of different datasets and offer datasets or the metadata (using VoID descriptions for example [Alexander et al. \(2009\)](#)) in the dataset as a starting point. A majority of browsers we surveyed are prototyped over a single dataset because of scalability issues, especially in cases where the data browser uses advanced query features (such as SPARQL) which are available in single datasets contained in one database. In accessing datasets or data sources, we identify three approaches which are commonly used by data browsers as exploration starting points:

Using URIs One approach to start an exploration in a data browser is to provide a URI, as an analogy to a URL entered in a Web browser. The data retrieved about that resource is displayed in the browser and used for subsequent navigation and browsing. The concept of using URIs in this context might be foreign to most end users, since most users associate a URI with a link to a web page rather than an identifier denoting a real world object.

Keyword search Another common approach is to use keyword search over a dataset as a starting point. Unlike keyword interfaces in search engines or natural language interfaces, keyword search across browsers is rarely used to formulate the full query or intention of the user, but rather to find entry points in the data, therefore replacing knowing specific URIs as a requirement to start browsing. Keyword search is usually performed either with the goal of identifying specific resources (e.g. a specific Country such as “UK”) or to find a particular collection of resources by finding their class or type (e.g. Countries). Once selected, the single or multiple resources are the initial set of resources from which exploration and browsing is further conducted.

Class hierarchy A third alternative is to provide users with browsing the class hierarchy of a dataset. Classes are useful descriptors about the content of a dataset (they provide users with a list of the different types of available information resources - e.g. Countries, Politicians, Cities etc.), and some interfaces provide hierarchical browsing through the class hierarchy, if one is provided in a dataset. This allows users to familiarise themselves

Tool	Starting point
Tabulator	User entering a single URI.
VisiNav	Keyword search usually for finding instances or classes where to start with navigation. It can additionally provide more structured search through drag and dropping URIs into a search bar (e.g. drag foaf:person URI identifier to retrieve all the people).
Humboldt	A list of all resources that can be aggregated by their classes (i.e. types).
Parallax	Keyword search over individual resources (topics in Parallax terminology) and or classes (collections of resources in Parallax).
Explorator	Keyword search for individual resources or set of resources from a certain class.
BrowseRDF	None described. Authors presume that initial set of resources are from a certain class.
gFacet	Keyword search over classes.
tFacet	Selecting a class through browsing the class hierarchy of a dataset.
Falcons Explorer	Keyword search for individual resources or set of resources of a certain class.
Sewelis	Selection from classes or using a constrained query-completion language.

Table 3.1: Exploration starting point implementations across different browsers.

with the content of a dataset, and is usually supplemented with a keyword search as well.

Table 3.1 shows approaches to exploration starting points across different browsers.

3.4.2 Navigation and Browsing

Once the initial set of resources have been discovered, all browsers rely on navigation through the links in the graph to explore and browse related data. We identified that navigation facilitates two functions; first it allows a user to explore and learn about unfamiliar datasets and second, query formulation in the interfaces is conducted by navigating and creating a trail of exploration. In the following, we examine different aspects of navigation through graph data in various browsers. Table 3.2 shows approaches to different aspects across browsers.

Browsing Browsing in the majority of browsers is implemented through pivoting i.e. refocusing from an initial set of resources to another set of resources via a common link. The links available for navigating from the initial set of resources is a union of the properties for each of the resources currently under inspection, and are presented to users as options through which they can pivot. For example, we might start from an

Tool	Browsing method	Bi-directional navigation	Branching	Relationships finding
Tabulator	Query-by-example	Yes	Yes	No
VisiNav	Pivoting	No	No	No
Humboldt	Pivoting	No	No	No
Parallax	Pivoting	No	No	No
Explorator	Finding links in sets of resources	Yes	Yes	Limited to one hop away
BrowseRDF	Pivoting	Yes	Yes	No
gFacet	Pivoting	No	Yes	No
tFacet	Pivoting	No	Yes	No
Falcons Explorer	Pivoting	No	Yes	No
Sewelís	Pivoting	Yes	Yes	No

Table 3.2: Browsing methods supported in different browsers.

initial set of “Countries”, and navigate through the “contains_city” property to find all of the corresponding “Cities” for which there is available data (Figure 3.4). Once we get related data about “Cites”, we can pivot again to find those which have hosted “Olympic Games”. This approach is used by the majority of browsers such as Parallax, VisiNav, Humboldt and gFacet. While using the same fundamental technique to navigate through data, the implementation metaphor across browsers is significantly different (Figure 3.5). For example, Humboldt looks like a standard faceted browser that allows refocusing on a selected facet. gFacet is a faceted browser visualised as a graph. The tFacet interface uses pivoting through a hierarchical-like interface in order to select sets that will be used as facets on the initial set of resources. Sewelís also follows a similar paradigm, where selected facets are used in combination with a constrained-language query and selection lists. These browsers try to extend a hierarchical faceted browsing paradigm over RDF data as a way of browsing a graph. Set-oriented approaches, on the other hand, distinguish facets for filtering and connections to pivot other sets. Immediate links can be used to filter the current set of resources or to navigate a related set of resources. An example of set-oriented approaches are Parallax and VisiNav.

Another approach to use pivoting is to navigate a graph starting from a single resource, and then use the navigation trail to formulate a query i.e. use the trail as a *query-by-example*. This is the approach taken by Tabulator and is based on navigating a single resource at a time. Once a path with a single resource is defined a user can then select and define a pattern by selecting specific properties and values as constraints in a query template. The overall results are then tabulated and presented in a table. One notable exception to the pivoting paradigm approach is Explorator, which allows users to instantiate multiple initial sets of resources. The Explorator interface then allows these resources to be linked if properties that link the individual items exist between

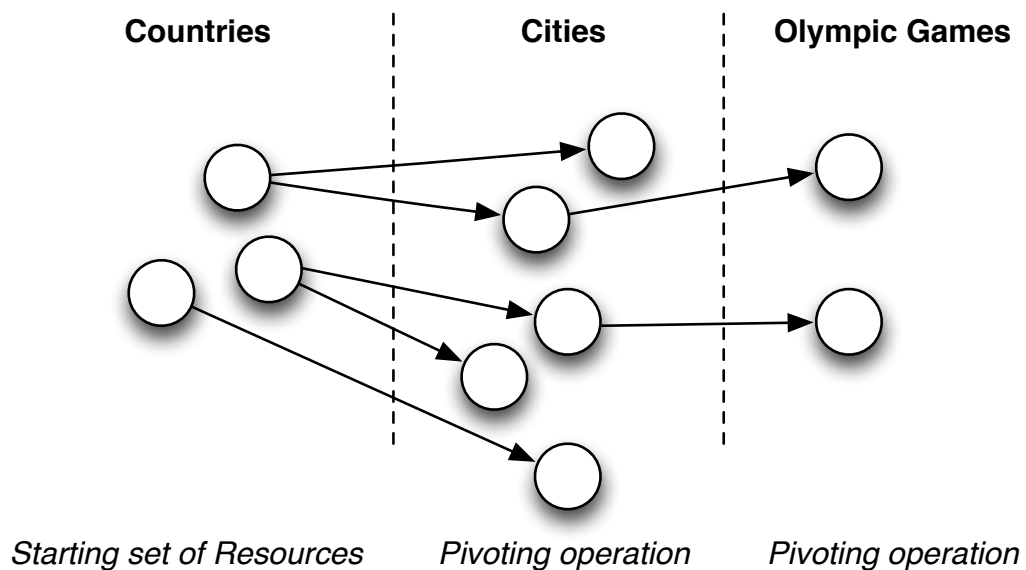


Figure 3.4: Pivoting with sets of resources.

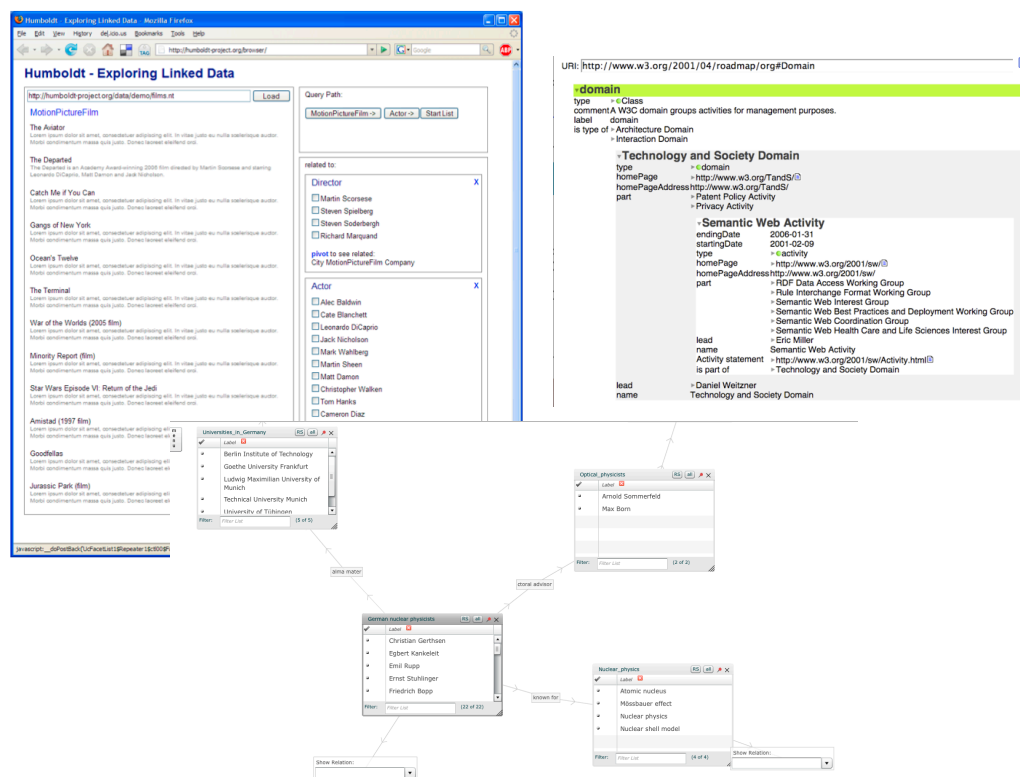


Figure 3.5: Tabulator, Humboldt and gFacet showing different data browsing visualisations

them; however the user interface only searches for the existence of direct links between sets of resources.

Directionality Since links in a RDF graph are directional, a browser can support browsing either by only using outgoing links from sets of resources or include back-links - links that other resources have to link to the current set of resources in the browser. Including back-links is important because they can increase the browsers expressivity. An open question remains however if the directionality of links needs to be explicitly stated to users or they can simply be treated equally as normal links.

Branching Data browsers allow users to navigate from several data resources to several other resources using common properties. However, given a set of resources a browser may allow users to choose a several of the offered properties to branch the current exploration, thus effectively instantiating several avenues of exploration at the same time. The process can then be repeated in each of the resulting sets of a branch. Figure 3.6 illustrates the concept of branching. The example shows that a data browser starts by focusing on a set of “Countries”. From there two pivoting operations are undertaken: one finds all the corresponding “Politicians”, while the other finds all the corresponding “Cities”. Then another branch is generated after two pivoting operations are taken from “Cities” to “Subways” and “Olympic Games”. Branching is important since it allows better expressivity in querying: in our example, if we were to ask for “All the politicians from countries whose cities have hosted the Olympic Games?” we would have to use branching to formulate that query.

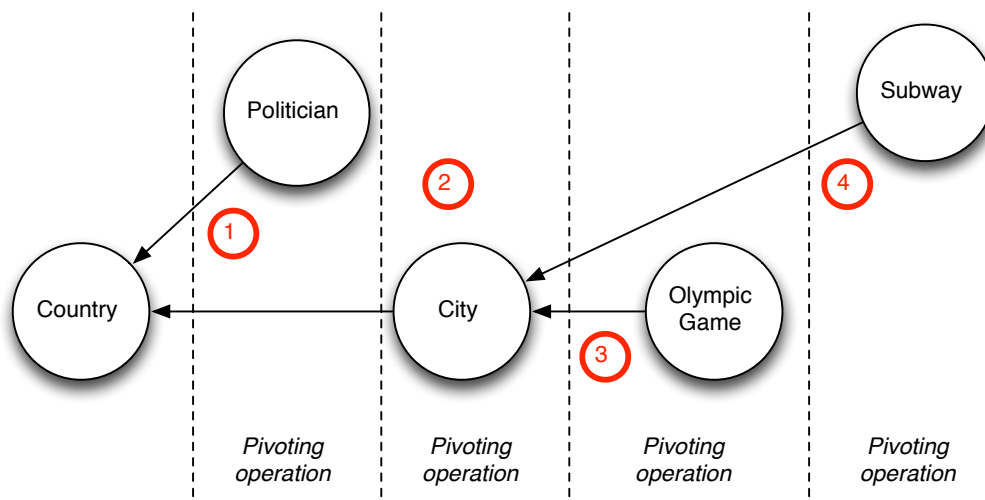


Figure 3.6: Example concept of branching in a generic data browser.

Thus, from our surveyed browsers we can distinguish tools based on the numbers of trails they can support. A trail is a sequence of pivoting operations, where every pivoting operation has exactly one predecessor and one successor. In principle, browsers can:

- Allow users to follow one trail at a time only. Some pivoting interfaces currently support navigation through a single trail with the ability to move back and re-direct the current browsing trail. Examples of such browsers are Parallax, Humboldt and

VisiNav. Each of them provides hierarchical breadcrumbs to navigate between pivoting operations. Whenever a new pivoting operation is initiated from a set of items in the middle of the trail, the previous trails gets discarded.

- Allow users to follow multiple trails at a time. Browsers can allow users to conduct multiple pivoting operations from a single set of items. This is, for example, the case with the gFacet interface, which combines faceted browsing with a schema-level graph visualisation. tFacet and Sewelis allow branching in the used hierarchical navigation to select facets for the initial records. Tabulator also allows multiple trails of navigation when browsing a single resource.

Finding relationships

As we notice, all of the browsers currently rely on navigation to find related data in a graph of data. This approach may be suitable for information seeking when we are browsing serendipitously for data, or when we already know how the schema of a database works. However, if the schema level information is unknown to a user, it may be a daunting task to rely solely on navigation to connect to data that can potentially be multiple links. One of the approaches to this problem is having the user interface support finding relationships between remote resources. The RelFinder interface, for example, allows finding relationships between several individual resources, but it does not support relationship findings on sets ([Lohmann et al. \(2010\)](#)). The interface might be adapted to enable a query-by-example system similar to Tabulator; however then the assumption is that users will have to somehow have prior knowledge of instances of a particular set.

3.4.3 Data Representation

Since RDF is strictly machine readable, it is up to the data browser to figure out how data is going to be represented in the browser. Generic browsers that deal with sets of resources need to support the representation of individual RDF resources as and representations of sets of RDF resources for their set-oriented paradigm. Table 3.3 shows how data is represented throughout browsers.

3.4.3.1 Representation of individual resources

Most of the generic data browsers use a single RDF resource as the smallest resource of information to be displayed in a browser. While this comes as a natural choice since each RDF resource should identify a recognisable real-world object, there are issues on how to best represent them. Representation approaches vary from browser to browser and usually depend on whether the representation is suitable for the specific purpose of the application. Common approaches are to display RDF resources by using its label, or

Tool	Individual resources	Data representation/interaction model
Tabulator	Labels	Resource and properties are represented in a nested table. Opening each resource creates a nested box of another resource.
VisiNav	Labels	Current focus are sets of resource with multiple views as a list or tabular representation.
Humboldt	Labels	List of the set of resources with side facets which can be used to pivot and refocus.
Parallax	Fixed Lenses	List of the set of resources with side facets for filtering and connections which are used to pivot to the next set. Alternative visualisations are available, such as tables, maps and charts over the current view.
Explorator	Labels	Multiple lists on the same screen can show multiple resources and are expanded to show an individual resource. Users combine the list with a set of operations in a side bar to create new set of resources.
BrowseRDF	Labels	Faceted browser with side facets that can be configured. The result set is a list of resources.
gFacet	Labels	A faceted browser where the facets are represented in a graph based on the navigation path. New facets are created by adding a link from an existing facet. Users can specify which of the facets is a result set, in which case all other faceted choices are used to filter the result set.
tFacet	Labels	Hierarchical navigation to select facets form an initial set shown as a table. The result set view allows different facets to be added as columns in the table.
Falcons Explorer	Labels	The initial set of items are shown as a table where properties from the set can be added as columns. To pivot, a user selects a column which in turn refocuses to a new table with the items in the selected column as a new focus.
Sewelis	Labels	A simple faceted browsing interface with a list result set and list of facets. Additionally a textual representation of a query can be used to show the context of the exploration path.

Table 3.3: Data representation across browsers.

using heuristics by testing for additional literal properties that are suitable for resource representation. Other more generic approaches include representing a resource by listing all of its properties, although this approach is often not taken when an interaction in the interfaces is done with sets of resources. In some datasets (e.g. in Freebase’s Parallax) an individual RDF resource can easily be mapped to a representation because certain properties are assumed for every single resource (e.g. a label, abstract, and depiction property). However, this is not the case in all RDF data that exist out in the wild.

Lenses - visualisation templates about RDF resources - can lead to increased usability and accessibility of data in the browser; however it might be disruptive to the generic approach of the entire interaction model of an interface which is based on a single RDF. In our example data, an “Olympic Game” resource can be represented using values of resources several hops away from the resource it describes (e.g. using data about the Country, for example). If these are surfaced in the browser when pivoting is implemented on the RDF resource level this can lead to the display of several sets of things into focus and thus can be confusing when subsequent pivoting is engaged. For example, if “Olympic Game” resources show information about the “Country”, then a pivoting operation to get the “Countries” through “Cities” is redundant. Such problems make it hard to bring lenses to generic data browsers. Additionally, lenses need to be published by some party, either the original data publisher or another party, which places an additional burden on the already published raw data. Therefore, we rarely see the use of lenses in generic data browsers. This problem of representation of individual data sources is the reason we can see a big regress in usability and overall user experience when comparing configured user interfaces (such as Parallax) to more generic interfaces e.g. Tabulator. Thus, adopting minimal conventions for describing RDF resources, such as the OpenGraph²¹ protocol does (e.g. mandate basic properties such as labels, depictions, descriptions and types for all resources) can result in increased usability of data displayed in generic data browsers.

3.4.3.2 Representation of sets

Since pivoting assumes navigation from one set of resources to another, browsers need to represent the plurality between pivoting operations in order to provide context between multiple pivoting operations. Representation and visualisation between multiple related sets of resources, however, varies between browsers depending on the interaction model and the purpose of data browser. In this respect, we made a distinction between two types of browsers:

- **Browsers extending a faceted browsing paradigm.** Browsers that use a faceted browsing metaphor over graph data use pivoting navigation in order to

²¹<http://ogp.me/>

browse the graph for suitable facets on the initial set of items. Additionally, these interfaces allow refocusing i.e. each of the facets can become the result set of the faceted browsing interface. Since these browsers put forward a faceted browsing paradigm over graph data, visualisation of plural relationships between sets is not shown - rather relationships can be viewed by filtering the faceted browsing interface. For example, a browser can start from “Olympic Games” and navigate to “Cities” and “Countries” and select these sets or some of their literal properties as facets to “Countries”. Thus, if a user wants to view which of the Olympic Games has been hosted by a particular country (e.g. the UK) the UK facet can be selected to filter for the corresponding Olympic Games.

- **Browsers supporting visual representations.** Browsers such as Parallax and Tabulator allow navigation through the graph and support visualisations that show the context between multiple navigation steps. In Tabulator, this is facilitated through a nested visualisation of resources or tabulation when multiple resources are tabulated. In Parallax, after a pivoting operation each set is replaced on the screen with the resulting set. In order to view relationships between items of two related sets, each new item in the set contains a header that shows its relation with items in the previous set. However, the context of the first set will not be viewable if another pivoting operation occurs, since Parallax only shows this context between two “neighbouring” sets. Alternatively, Parallax allows users to create tables with selected columns by specifying a pivoting trail (Figure 3.7). Because items can exhibit many-to-many relationships the table is not grid-like but irregular - a cell in a column can hold multiple values.

Another aspect of representation is how navigation trails are represented in a browser. As we’ve seen, browsing through data can produce multiple browsing trails in the graph. These can be quite complex trees that might need to be looked jointly to perform a specific query - thus in viewing this context supporting a quick refocusing on different parts of the trail is important. In our example, if we start from “Countries” we may need to navigate to “Cities” and “Olympic Games” if we need to answer queries such as “Which countries with Cities that are under a million have hosted the Olympic Games?” Thus we may be required to go through the different portions in our exploration to select or filter for particular bits of data. Browsers support this in a variety of ways. In Parallax and Humboldt, for example, since only navigation along a single trail is allowed, navigation steps taken by users are represented with simple breadcrumbs. In gFacet the entire faceted interface is laid out in a graph, therefore showing the entire context of the exploration. In Sewelis, no browsing trail is shown - rather the query completion interface can be used to view and refocus on a particular part of a query.

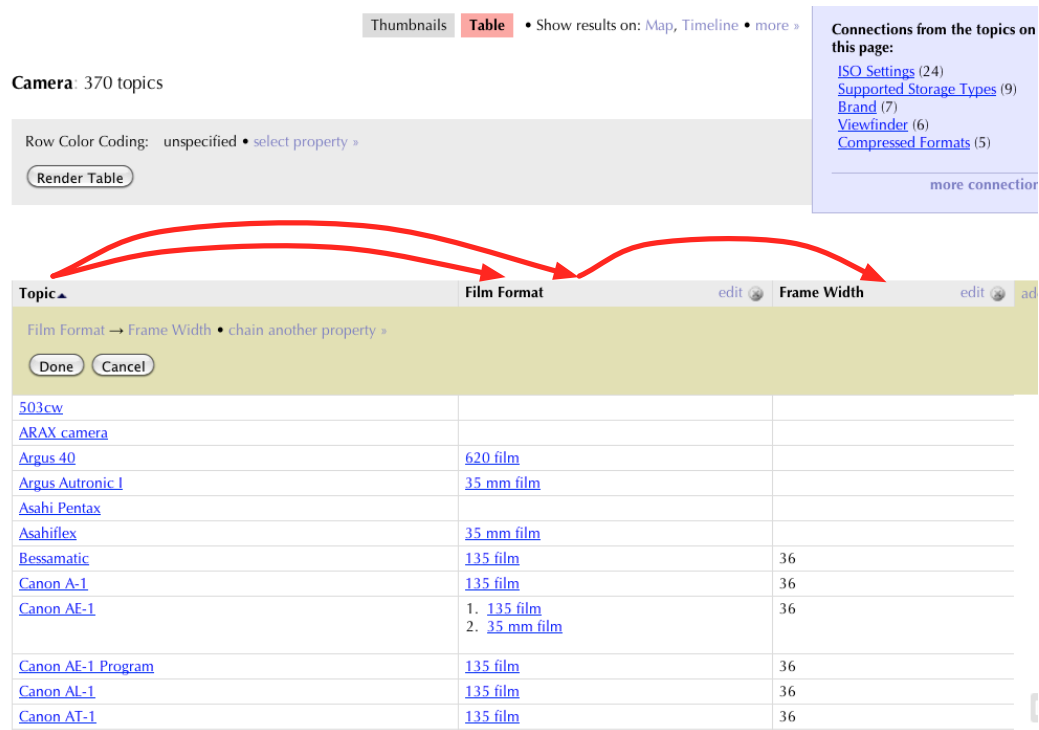


Figure 3.7: Parallax showing multiple navigational trails displayed in a spreadsheet.

3.4.3.3 Visual aids and tools for analytics

Most browsers focus solely on exploration and querying data. Some browsers, however, provide additional features that not only allow browsing and querying data, but also data gathering for the purpose of visualising, re-representing or analysing data, therefore supporting different views, rather than just resorting to the generic representations of raw data offered by a browser. Only two of the tools in this survey support such features - Tabulator and Parallax. Tabulator allows users to represent tabulated data queried through its interface using a variety of widgets that support different views: charts, maps, timelines, and calendar views. Parallax offers similar extensions. A depiction of visualisation widgets in both interfaces is shown in Figure 3.8.

3.4.4 Query and Filtering

3.4.4.1 Filtering across trails of navigation

Distinction between browsers can also be made based on how they allow users to filter data. Filtering in this case is examined based on how filtering is propagated in navigation trails. As we already know, generic browsers such as Humbolt, gFacet, BrowseRDF, tFacet and Sewelis aim to extend faceted browsing to RDF data. Therefore, filtering for

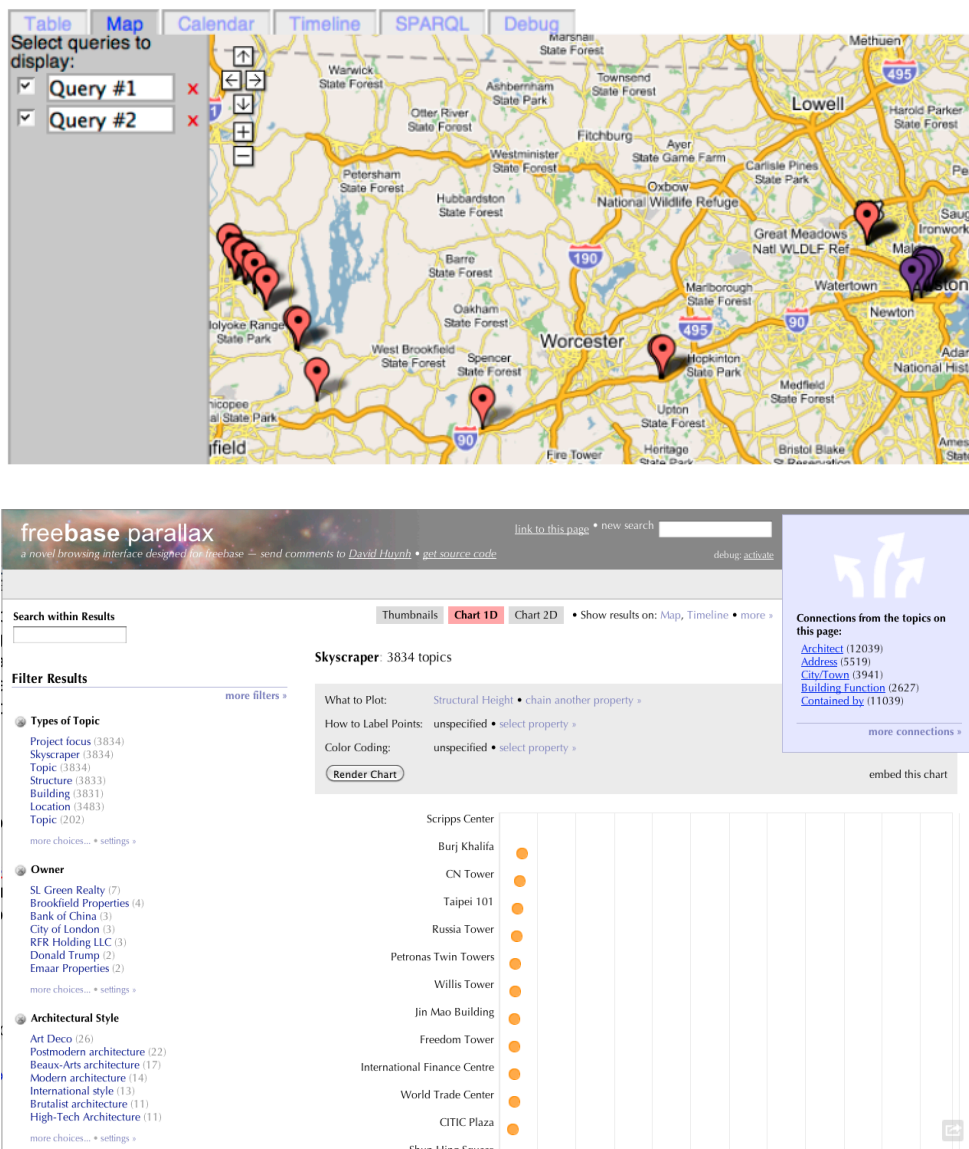


Figure 3.8: Visual aids offered in Tabulator and Parallax to visualize and analyse collected data.

data follows the same pattern as any faceted browser with hierarchies of facets. On the other hand, set-oriented approaches distinguish between pivoting as navigation among related sets of resources, and filtering through facets, which can filter the current focus set of resources in the browsers based on the resources attributes. For example, in Parallax and Falcons Explorer, filters over trails of navigation apply unidirectional i.e. if a set is filtered in the middle of a browsing trail the results of the filtering are only reflected in the direction of the trail. In order to filter in the opposite direction, these browsers need to start from the set of resources and re-navigate to the previous initial set.

Tool	User studies/Evaluation Method
Tabulator	None
VisiNav	Small descriptive “loud thinking” user studies during development. No comprehensive studies with final interface.
Humboldt	Small-scale usability study.
Parallax	None
Explorator	Pilot study and a small-scale experiment.
BrowseRDF	Formal evaluation of interface by comparison with other faceted browsers and a user study. During the study users were given the schema of a dataset to help them formulate queries.
gFacet	Comparative task performance evaluation with Parallax.
tFacet	None
Falcons Explorer	None
Sewelis	Usability evaluation with 20 people over a variety of different tasks with increasing level of difficulties in a number of categories.

Table 3.4: Evaluation and user studies made on Linked Data browsers.

3.4.4.2 Additional filtering options

In addition to selecting filters through facets and pivoting from one set of resources to another, data in browsers can provide additional filtering options. Examples mostly include adding Boolean and set operations to the query interface. For example, BrowseRDF allows a selection based on existential operators. Both BrowseRDF and Sewelis allow negation operators. Explorator is built on the notion of instantiating different sets of attributes and combining them with set operators such as intersection to find common resources. [Hearst \(2009\)](#) notes, however, that such advanced query operations have often been found to be unusable and have not seen wide adoption.

3.4.4.3 User Studies and Evaluation

The final step of our analysis is to investigate if and how these browsers have been evaluated, if any of them were submitted for general usability testing, or if the evaluation targeted a proposed design around a specific problem. The evaluation undertaken by browsers is shown in Figure 3.4.

3.4.5 Observations and Conclusions

As we can observe from our analysis, numerous attributes need to be taken into consideration when designing data browsers. From the browsers described in the survey, we can notice that design implementations between browsers differ extensively. However,

one might also observe general commonalities between data browsers. First, we notice that all browsers use minimal generic data representations, and except in special circumstances, they never try to include richer representations using, for example, lenses. In the analysis, we gave both design and economic reasons behind such decisions. This point, however, suggests that a (truly) generic browser can only provide more raw data access, and thus its designs should be aimed at audiences that are able to handle data in a more raw format - users such as Chris - the user in our first persona. Second, based on the overall design and stated motivation, we find that we can group existing browsers into tools that provide:

- **Grafting faceted browser interfaces onto graph data.** All of the interfaces in the survey are motivated by the fact that browsing graph data is inherently hard in the absence of any technical knowhow. More than half of the approaches, however, are designed with the purpose of facilitating exploratory search and/or query answering over large graph datasets by adapting established models of data interaction on RDF. As we've seen, browsers such as gFacet, tFacet, Humboldt, BrowseRDF, and Sewelis focus around grafting faceted search interfaces onto arbitrary graph datasets. Thus, the concept and interactions of browsers revolves around specifying a result set and facets to be used for filtering. Thus, this interaction can effectively be seen as end-user created faceted browsing interfaces over RDF on demand.
- **Extending faceted browsing paradigm.** Another class of Linked Data browsers are those that extend the faceted browsing paradigm to data containing multiple entity types. Additionally the purpose of these tools is not only for exploring and answering data-centric questions, but to provide an interface for mashing and presenting data from multiple sources. Thus these browsers tend to resemble mashup tools as opposed to just faceted browsers. As such, the metaphors around how the data is browsed, queried and represented are done with that specific purpose in mind. For example, browsers such as Parallax and Tabulator use pivoting (albeit in different ways) to find and gather data, both allow querying of data and tabulating them with visualisation widgets to represent and further analyse the data in other, useful representations. Similarly, while Falcons Explorer does not include data visualisation tools, its envisioned use is a tabular end-user programming paradigm.

Given these two categorisations, we can notice that only Tabulator and Parallax are designed with the motivation to provide interactions over Linked Data similar to the ones described in our first persona.

3.5 Prototyping a Generic Data Browser - The GEORDi Experience

The design process described in this Chapter attempts to suggest challenges in end-user interactions over Linked Data by going through a requirements elicitation exercise. It first examined the activities end users engage with when dealing with typical data-centric interactions, and provided examples where the Web falls short in providing data-centric capabilities. Second, it examined the functionalities offered by current generic data browsers. We noticed that out of ten browsers, only two were designed with the motivation to provide exploration and sense-making over Linked Data in a way described in at least one of our personas. Given the number of functionalities that might impact a design of a generic data browser, we might ask: Are the proposed functionalities in the current set of generic browsers suitable to address all the challenges end users face when accessing Linked Data? What areas of end-user access to Linked Data are challenging and require different solutions?

In order to identify these areas, we present the results from a prototyping exercise that included designing and implementing a generic data browser, named GEORDi, around the requirements stated in our first persona. The reason we choose to build our own generic browser instead of using existing ones is that none of the current set of browsers fully satisfies the requirements for accessing datasets in a way described in our first persona. As we noted in the previous section, most generic browsers are designed as interfaces enabling faceted browsing over Linked Data on demand, which satisfies only part of the overall process of data-centric interactions as described in the scenario. For example, most of them did not use a tabular or other visualisation that resembles a spreadsheet, a metaphor around which we wanted to build our interface, nor did they provide any tools for data analytics. The two generic browsers that are designed to support data-centric interactions similar to those described in the scenario, Tabulator and Parallax, have significant drawbacks. The first, Tabulator, is a query-by-example interface, which represents a step back in usability when compared with current advanced search interfaces that offer querying functionalities through interfaces such as faceted browsing. Additionally, its reliance on dereferencing URIs for querying is very limited, as more interactive exploration over data requires database capabilities. Parallax, on the other hand, represents a hybrid interface, which seems to be tailored for users both with and without basic end-user data manipulation skills. Additionally, Parallax is built specifically for the Freebase dataset, and thus relies on several underlying assumptions which are specific to the Freebase dataset. For our prototyping exercise, we need a browser that can access various data repositories on demand.

In the following sections, we describe (1) the design rationales and system description of GEORDi, and (2) observations and identified challenges from internal testing of GEORDi over several Linked Datasets.

3.5.1 Design Goals

GEORDi design goals were motivated by the scenario we described in our first persona i.e. supporting end-users with experience in end-user data tools, such as spreadsheets, in exploring and interacting with data from various Linked Datasets published on the Web. Given our first persona scenario as a template, GEORDi was designed along the following goals:

- **Be able to open different datasets on demand.** Our first design decision was to be able to access numerous Linked Datasets on demand. Given the requirements for accessing and filtering through multiple data entries, a generic browser will need to use advanced interactions that require database capabilities. Thus we only included datasets that are accessible through a live SPARQL endpoint.
- **Represent data and data browsing using a spreadsheet metaphor.** Spreadsheets are a WYSIWYG interface - people can perform entry of data, interact with data and visualisations directly from the spreadsheet representation of the data. Thus, in order to leverage familiarity with spreadsheets by using spreadsheets as a metaphor, we decided to implement both data representation and browsing to be as closely associated to spreadsheets as possible.
- **Use set-oriented browsing.** Since the scenario described includes finding and integrating data about multiple data resource simultaneously, we decided to utilise set-oriented browsing in our browser.
- **Include tools for visualisation and analysis.** The use of spreadsheets was also motivated by the fact that an easy to use spreadsheet visualisation of graph data can be easily visualised in other formats. Thus we decided to include simple chart and map visualisations as part of our generic browser.

3.5.2 System Description

In this section we describe the GEORDi interface in more detail. The descriptions focus on the key features of the systems: (1) dataset discovery, (2) data representation and browsing, and (3) advanced representations of linked data through lenses and user generated visualisations.

3.5.2.1 Exploration Starting Point

In GEORDi users can discover potential Linked Data sources either through a catalogue of datasets that GEORDi knows about, or alternatively use keyword search that returns

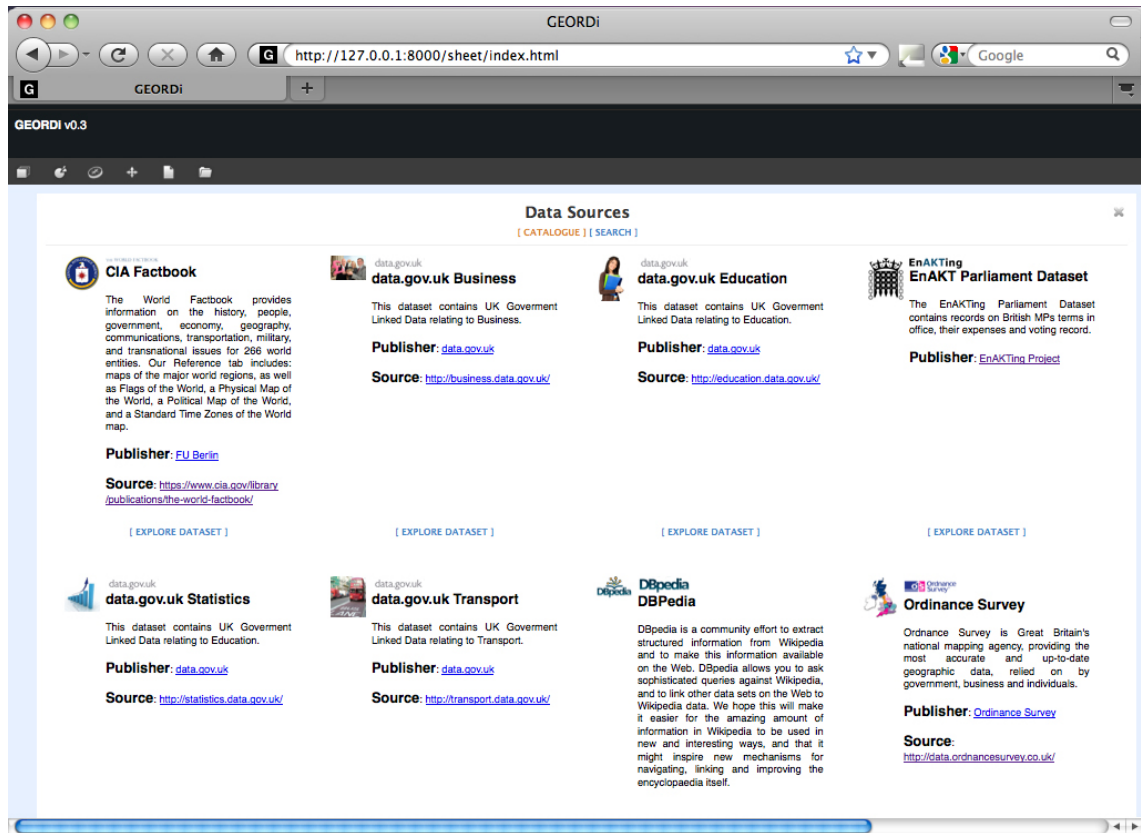


Figure 3.9: The GEORDi catalogue of datasets.

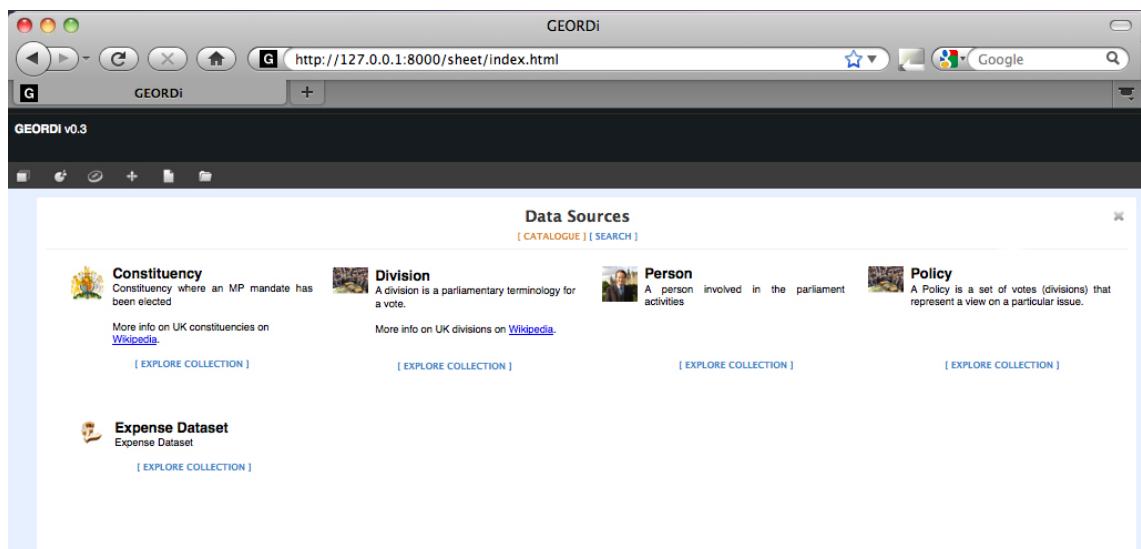


Figure 3.10: Collections contained within a dataset represented in GEORDi.

associated RDF resources, classes or datasets. Alternatively, if a SPARQL endpoint URL is known, it might be inputted and added to the catalogue.

Figure 3.9 depicts the catalogue in the early GEORDi prototype. Each entry in the catalogue holds the name of the dataset, a short description of the data one might find in the dataset and some additional provenance information, like the institution or

person publishing the information and a Web site where the user can refer to for more information about the publisher. Dataset information was easily derived from VoID descriptors of the Linked Datasets. While the information is a few short lines, it serves as a preview or a cue to users about the sort of data they might find in a particular dataset, so they can decide whether it is worth diving in a particular dataset to search for any useful data.

If users decide to explore a dataset, they can simply click the “*Explore Dataset*” button which yields a list of classes defined in a dataset. To use a more user-friendly wording, in GEORDi these are named as *collections*, because they provide groups of resources of a particular item. Collections serve as a starting point from which the user can start exploring the actual data. Similar to the data catalogue each collection shows a brief description. Figure 3.10 shows the collections contained in the “UK Parliament” datasets. As the Figure shows, a user can start exploring these datasets by opening the Constituency collection, Person collection etc.

The catalogue and collection mode allows users to browse through datasets whenever their search is of an exploratory nature. GEORDi, however, allows users to also do a keyword search which returns relevant RDF instances, collections or datasets.

GEORDi is implemented in such a way that each dataset actually corresponds to a single public SPARQL endpoint, and the collections in each dataset represent a determined set types of classes from which the user can start exploring the graph. While data from multiple datasources can be opened simultaneously, these cannot be easily combined and queried jointly because of the limitations of current SPARQL stores. This might be available in the future, when more mature frameworks offer the possibility of responsive, federated queries on demand.

3.5.2.2 Data Representation and Browsing

Data described using the RDF model include many-to-many relationships. Relational databases deal with many-to-many relationships by using multiple standard tables (i.e. tables that have equal number of cells in each row and column). Relationships between these tables are established by using primary and foreign keys (Codd (1983)). In order to represent many-to-many relationships in GEORDi, we chose a simpler method by using irregular (nested) table representation of graph data. Figure 3.11 illustrates this concept by visualising sample data from the example represented in Figure 3.3.

Through the nested table representation, GEORDi implements the set-oriented paradigm to provide maximum context for the entire trail of pivoting operations. The representation also allows branching of exploration trails. Once a user finds a dataset for exploration and chooses a particular collection to start exploring, selecting that particular collection instantiates a list showing all the resources or instances of that collection

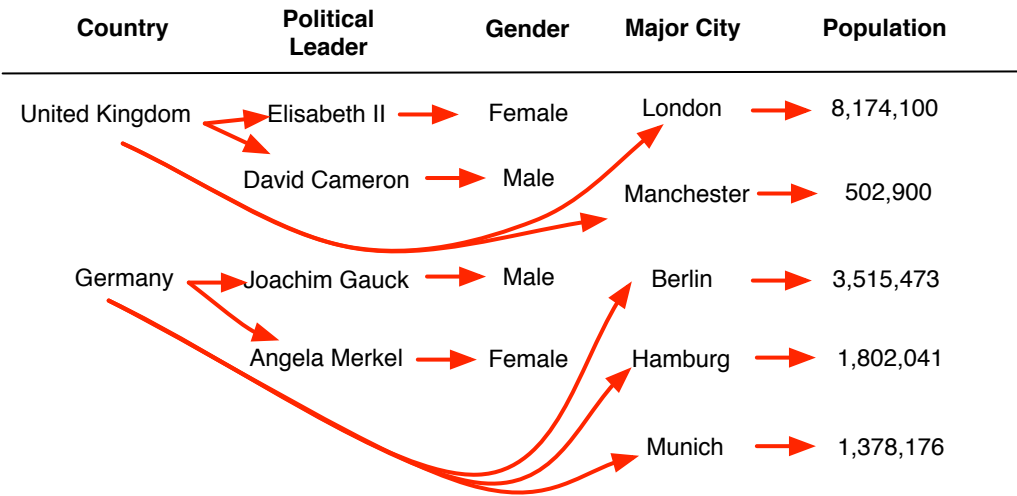


Figure 3.11: Concept of an irregular table to visualise graphs of data. The Figure shows entities described in cells. The red relationships illustrates how the entities are linked; the type of relationship is indicated in the column header.

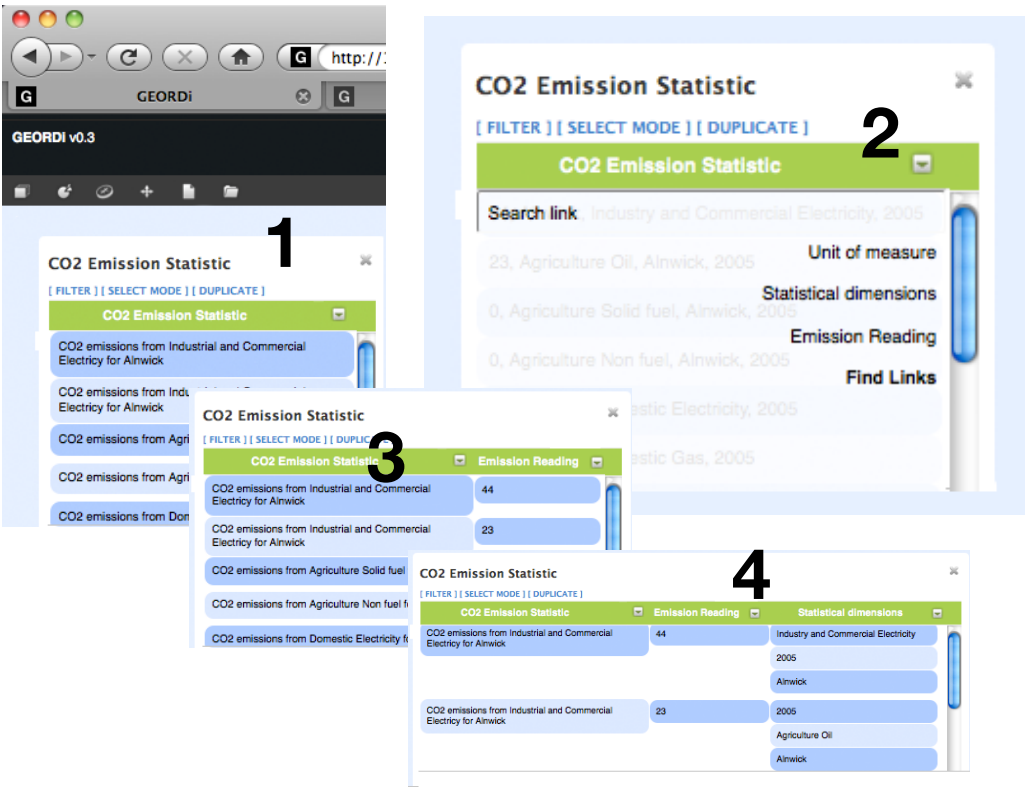


Figure 3.12: An example showing set-oriented operations in GEORDi over statistical data of UK government linked datasets. Users can open up new columns from any existing column.

(Figure 3.12 (1)). The header of the list contains a “property slider” that, if selected, displays a collection of the properties about those resources (Figure 3.12 (2)). This

collection represents a *union* of all the properties of the RDF resources shown in that list. Selection of a particular property in the menu produces another resource column that is appended to the initial column. The resources of the new column represent corresponding resources that are linked from the resources in the initial column with that property (Figure 3.12 (3,4)).

Thus data-exploration in GEORDi is effectively represented as a spreadsheet generation, where a user generates new columns with data through multiple link-slides. The nested table representation is calculated in such a way that the height of a single resource cell is equal to the maximum height of all the resources which have been derived from that resource by pivoting. As with the initial column, the user has the ability to pivot from any other column, therefore to slowly unpack the graph by building up a custom spreadsheet. As mentioned previously, the users can view the entire context of their pivoting operations, thus allowing them to view relationships between items beyond successive pivoting operations only. In addition to generating a spreadsheet out of Linked Data, the user can filter results of any of the column as shown in Figure 3.13.

GEORDi allows the user to instantiate as many spreadsheets as they like, either by reopening the catalogue and selecting another dataset or collection. Additionally, users can create duplicates of the current spreadsheet and then take to link-sliding across different paths allowing them to see the results of both spreadsheets side-by-side.

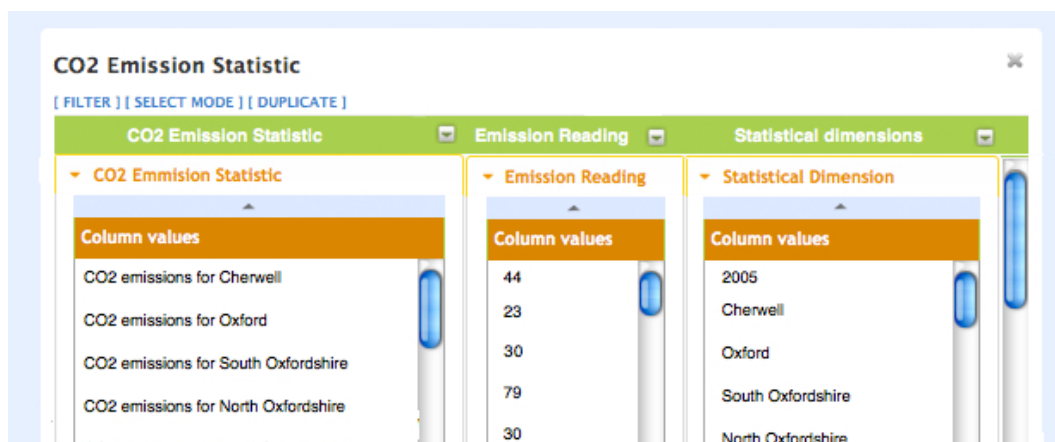


Figure 3.13: Filtering in GEORDi. The filters allow users to filter through the unique values of each of the columns.

3.5.2.3 Tools for Visualisations and Analysis

As part of the initial GEORDi prototype we included tools to visualise and analyse data that were shown in spreadsheets. This included visualising numeric and ordinal data in charts, time data in timelines and geographic points on a map. Figure 3.14 displays data from a spreadsheet visualised using a chart widget. While GEORDi allows exporting data as spreadsheets, which allows data to be used by more powerful and commercial

end-user spreadsheet tools, we included a set of simple visualisation tools in order to offer users a quick way to visualise data without requiring them to go back and forth a spreadsheet program.

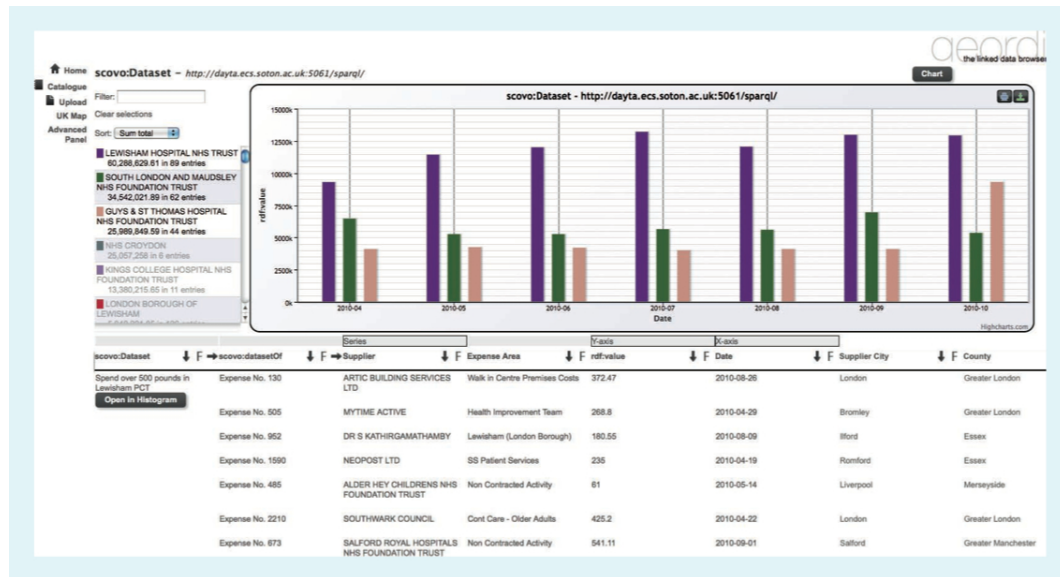


Figure 3.14: The final GEORDi prototype showing tools to visualise and analyse collected data.

3.5.3 Deployment and Observations

In order to ascertain if we could carry out data centric tasks as well as inspect the usability of GEORDi we undertook a series of preliminary informal tests by deploying GEORDi and accessing 16 different Linked Datasets. The goal of these tests was to observe and pinpoint specific problems in browsing Linked Datasets. In order to test various data-exploration concepts in GEORDi, our set included both datasets that were publicly available and were unfamiliar to us, as well as datasets we have previously published as Linked Data ourselves (described in Omitola et al. (2010)) whose structure was highly familiar to us. Our initial set of tasks included well-defined problems (e.g. “Find countries that only have cities with population over 3 million and do not have a subway system.”) as well as more open-ended tasks (e.g. “a freelance exercise to find interesting data about Countries in the CIA Factbook” or “What interesting relationships or connections can we find between countries and their political leaders”). Even during these informal tests several observations were immediately noticeable:

- Our most noticeable difficulty was in exploring unknown datasets i.e. datasets in which we had no knowledge of the schema. This proved particularly difficult in cases where the dataset contained many collections and graph data was highly dense i.e. the number of links between resources was high (e.g. the DB-Pedia dataset). Even with the ability to quickly iterate by opening multiple new

columns, we were overwhelmed when we were confronted with large datasets, such as DBPedia, which had numerous collections and property linking resources. This observation is also reinforced by the fact that some of the user studies over data browsers (e.g. [Oren et al. \(2006\)](#)) were designed in such a way that users were given schema level information during sessions in order to formulate queries in the interface. Additionally, formulating more exploratory queries, such as how do instances from one class connect with instances in another class that might not be immediately linked, was difficult when one was forced to choose a single starting exploration point in the browser.

- In addition to difficulties in finding and exploring large datasets only by navigation we noticed that our spreadsheet visualisation of graph data did not make it easy for us to understand the schema or ontology of a dataset. For example, following which column was derived out of which column was difficult to follow after ten or more set-oriented explorations.
- The GEORDi browser was designed as a single direction browser; it allowed pivoting by selecting one of many out-going links. We noticed, however, that several of our queries were significantly more difficult to answer because navigation was only permitted in one direction. In order to use the back-links, we needed to make two forward-link pivoting operations which added redundant data to our spreadsheet.

Our observations pointed out that a major problem of navigational generic browsers was the fact that none of them provided a way for users to effectively explore unfamiliar datasets. We concluded that exploring rich and complex Linked Datasets by navigation does not allow for efficient exploration of large unfamiliar datasets. Given that reuse of others data is among the most advocated reasons for publishing data on the Web as Linked Data, we concluded that interrogation of datasets required solutions beyond navigation as a tool to aggregate and query a dataset.

3.6 Summary

In this Chapter we described a design process intended to guide the design of data-centric interfaces over Linked Data. We proposed two scenarios that engage in data-centric interactions where combining data is the key to solving their tasks. Assuming that Linked Data can improve and make such tasks more efficient if proper interfaces for end-users exist, we elicited the required activities to accomplish these tasks, compared Linked Data alternatives and analysed if and how existing data-browsers support these activities. Our analysis points out that very few browsers are designed around specific use-cases. Additionally, few user studies are designed to test for specific novel interactions introduced in these browsers. Finally, through this analysis and the prototyping efforts in GEORDi

we pinpointed several challenges using generic browsers that are based on pivoting. The challenges from the design process can be organised along two major challenges:

- First, the design process described in this Chapter suggested two different personas, with different types of stakeholders engaging in data-centric interactions. We can notice, however, that most of the generic browsers we surveyed, as well as the GEORDi prototype, are by design, closely associated with the underlying triple data model of RDF. Even by using labels, or perhaps even employing lenses per resource level, these interfaces still expose the user closely to the underlying machine representational format. Thus, generic browsers such as the ones we described throughout this Chapter can only be usable for people with interest in interacting with data in a more raw format - for example users such as described in our first persona. In our second persona we described a scenario in which data-centric interactions were performed by users which do not have skills to handle raw data; rather they experience data only through the lens of an application. Thus our first challenge is how one can bring data-centric interactions closer to the Web experiences the users are accustomed to without having the data-silo attributes that are usually associated with Web applications.
- Second, our analysis showed that generic data browsers and the GEORDi prototype used navigation as a technique to both query and explore datasets. Preliminary testing of GEORDi however, suggested that data navigation and pivoting is not an effective technique to explore large unfamiliar datasets of Linked Data. Additionally, we found that existing representations of instance level data provided a poor representation of schema level information of a dataset. Finally, we also noticed that navigation or assembling query through the use of back-links was just as important as navigation provided through out-going links.

The following two Chapters describe solutions to these challenges. In the next Chapter (Chapter 4) we present Visor, a generic data browser that builds on our experiences using GEORDi and introduces multi-pivoting, an attempt to extend navigational interfaces to better support exploration in unfamiliar datasets. In Chapter 5, we present mashpoint, a framework that allows data-oriented applications to be linked based on the similarities of the entities in their data. By linking applications in mashpoint, we provide a way to support concepts like data navigation through links, which allows data-centric interactions to be performed without the use of generic data browsers. Thus mashpoint allows the Web to exhibit data-like interactions, making data-centric interactions accessible to more casual users.

Chapter 4

Multi-pivot Exploration of Data on the Web

Challenges in browsing large graph datasets are rich: large numbers of ontology concepts, and high entropy and diversity in links between individual data instances often make it hard to understand both the overall content of a dataset, as well as understand and find the particular bits of data that might be of an interest. Such problems can often overshadow the benefits of interacting over large highly interconnected data. The goal of generic data browsers has been, in part, to tackle the problem of making sense of such rich and complex data domains. As discussed in Chapter 3, a common technique that has been adopted by a number of data browsers for exploring large graphs of data is *pivoting*. In this Chapter¹ we focus on the limitations imposed by several commonly observed design patterns found in pivot-based data browsers: (1) exploration is often restricted to starting from a single point in the data, (2) navigation is typically supported in a single direction, and (3) immediate instance level exploration is regularly preferred without gaining familiarity with the domain or setting the exploration context first. This thesis argues that these characteristics impose a number of limitations: in the case of: (1) they reduce flexibility and therefore the ability to quickly find data that are related to the initial set multiple hops away; in the case of: (2) they unnecessarily reduce the expressivity of the browser, and in the case of: (3) the absence of an overview of the domain under exploration can often lead to difficulties in retracing exploration steps as well as make potential alternative exploration paths difficult to recognise. In this Chapter, we introduce a novel approach called a *multi-pivot* which extends the traditional pivoting techniques to mitigate the aforementioned limitations. We also describe a demonstrator tool named *Visor* which implements this approach and we describe the results of a user study to test the viability of this approach.

¹Parts of this chapter were included in a paper published at ISWC2011 (<http://eprints.ecs.soton.ac.uk/22784/>)

The outline of this Chapter is as follows. In the following section we discuss the multi-pivot approach, and lay out key design requirements for this approach. Section 4.3 describes Visor, a tool which was developed to test the multi-pivot technique. In Section 4.4 we carry out an evaluation study to test the viability of the approach and discuss implications for design.

4.1 Limitations of Navigation-based Browsers

In Chapter 3 we briefly described some of the limitations of pivoting as an interaction technique for exploring and interrogating unfamiliar datasets. In this section we describe these limitations in more detail. To better illustrate these, we consider a hypothetical situation of a data-finding activity based on the scenario that was used in the first persona described in Chapter 3. In that scenario the user had the task of finding data about countries. A particular part of the data-finding task was finding data about Olympic Games held in a selected subset of countries. In our example, we make the task more complex by adding additional data-finding tasks. For example, a data-finding task where the user needs to find all the Olympic Games, including the country they were held in as well as the people that performed the opening ceremonies. If we suppose that the user can answer this question using the DBpedia dataset, a user would have to find and query a particular part of the dataset (Bizer et al. (2009)). Figure 4.1a depicts the subset of the DBpedia dataset needed to answer the given query. To illustrate the challenges of finding this subset of data we consider the following statistics about the entire DBpedia dataset. The DBpedia² dataset contains 272 Classes (i.e. the DBpedia domain ontology), 8813 edges, 24448 datatype properties (links to literals), 627 object properties (links between entities). Figure 4.2, gives a visual representation of the complexity the DBpedia dataset³ and its relative size compared to the subset needed to answer the query. Given that users will often need to filter and use only a small subset of data from the entire dataset, a data-browsing interface needs to facilitate efficient interrogation of datasets in order to quickly ascertain whether a dataset has particular data in order to answer a data-centric question or not. To illustrate how navigation-based browsers limit efficient interrogation of graph datasets we examine the data-finding scenario through the attributes of navigational-based browsers identified in Chapter 3.

Exploration starts for a single point. As discussed in Chapter 3, in a standard navigation-based browser, exploration of a dataset begins with a particular set of instances. The initial set of instances usually pertains to instances from a certain class that is typically found through a keyword search or a catalogue of classes. In the example, users can start their exploration from either “Countries” or “Olympic Games. Once the initial instances are shown, users are presented with a number of properties which

²These were calculated in March 2011.

³<http://wiki.dbpedia.org/Ontology>

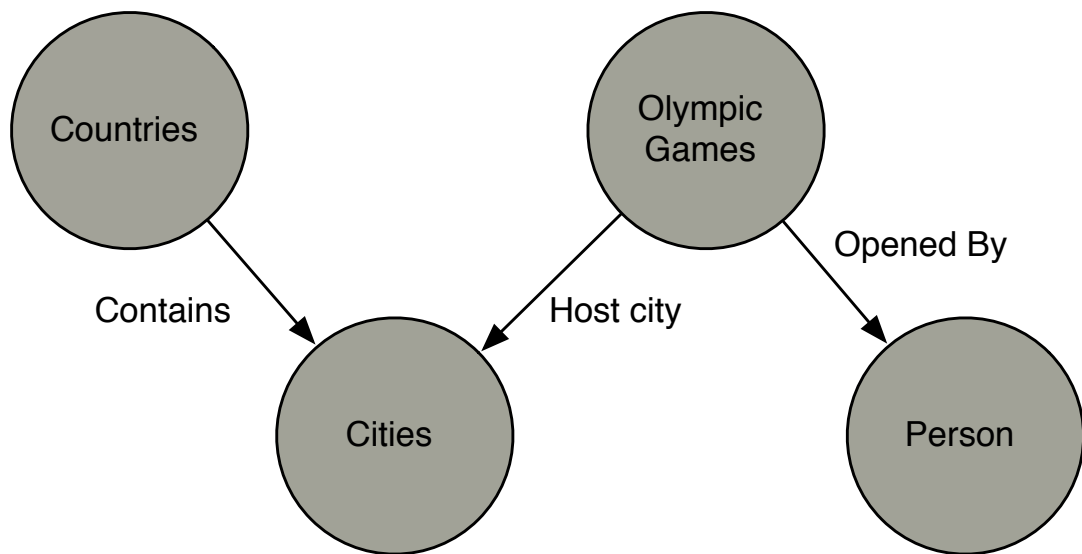
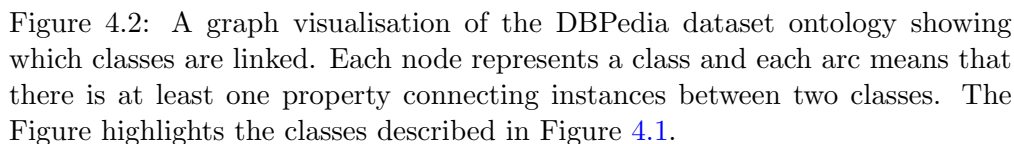


Figure 4.1: A portion of the DBpedia ontology showing the links between instances of the classes Olympic Games, Countries, Cities and Person needed to answer a query.

can be selected to navigate and simultaneously get all the instances which are related to the instances in the first set through that property. In such a way the graph navigation is facilitated. Let us suppose that either by keyword search or selection of classes, “Countries or “Olympic Games are among the choices for the exploration starting point. At this point users can choose to start with the instances of either class as their initial set. Let us suppose users choose “Olympic Games. As can be seen from Figure 4.1 users would need to perform two pivoting operations to get to “Countries. Since the two are not directly linked, users would need to do some exploring using navigation to find out the relation between “Olympic Games and “Countries. Unfortunately, no cues are given to guide users in which direction to start exploring since generic data browsers only show context and information about one navigation step ahead. In a situation where the domain is unfamiliar, this presents a problem. Property labels, which are used to show what is being navigated, do not hold any information about the path two or three arcs away of the current set of instances. The problem is further exacerbated when there are a high number of possible choices for pivoting and the number of choices increases exponentially if the relating instances of interest are multiple arcs away. For example, in the DBpedia dataset, on average, 32 property links are available for navigation, starting from a collection of instances of a certain class.

Navigation is unidirectional. As we pointed out in Chapter 3, the direction of pivoting in navigational-based browsers is often unidirectional i.e. navigation is enabled only with outgoing links from the instances in the current focus. This restriction can sometimes limit the query expressivity of the interface, or require the user to go through unnecessary navigational steps. In our example (refer to Figure 4.1) we notice that



Exploration and domain overview absence. Current data-browsers are predominantly instance-centric i.e. interaction is predominately over instance level data. Relying on instance data only to understand the overall structure of a dataset is often difficult, since representation of instance data is usually done in tables, whereas tree-like visualisations often result in a lack of overview about the sub-domain being explored as part of the exploration. The lack of overview can often lead to users missing unseen data relations and thus contribute to a lack of understanding about the domain being explored. For example, if the task in our example was more exploratory e.g. finding anything of an interest around Olympic Games, the user would need to quickly know about relationships between instances from different classes that might be several hops away in the graph and be able to scan multiple potential paths simultaneously. Research from the HCI community suggests that when confronted with complex information spaces,

information-seeking interfaces should follow the Visual Information Seeking Mantra: first an overview, zoom and filter, then details-on-demand (Shneiderman (1996)). The complexity and size of large datasets suggests that using such a paradigm might be suitable for data browsers.

4.2 Approach

In the previous section we pointed out several challenges to pivoting or navigation as an interaction technique for exploring graph-based datasets. In the following sections we introduce an approach that aims at mitigating these limitations. Based on the specific functions and interactions in data browsers that we identified as the causes of such challenges, we base the design of this approach around four general requirements (R1 - R4):

R1. Use multi-pivoting approach - navigation can be initiated by selecting multiple items of interest. Rather than being limited by starting with a single exploration point as in most browsers, our approach needs to allow users to start from multiple points of interest, and discover how the selected points of interest are connected to each other. In situations in which there are multiple ways that selected starting points can connect, the tool would need to provide users with the ability to find an appropriate path through a graph. An analogy to such a design would be a puzzle-solving example. When solving a puzzle, the solvers can start piecing the puzzle from multiple points: they can select several different pieces, find pieces that match, create several greater pieces and then piece these together to slowly gain an understanding of the overall picture. Similarly, our approach should let users grab different portions of the domain simultaneously, navigate either back or forth using normal links or back-links, build their own subset of the domain related to their interest. Since there was no central point where the exploration starts and users would be able to pivot freely from anywhere in any direction we named this approach a *multi-pivot* since it extends the standard navigational, pivoting approach⁴.

R2. Overview first, instance data on demand. The approach should not overburden users by immediately exposing instance data during exploration of the structure of an unfamiliar dataset. Rather, the approach needs to show navigation and exploration to users with information on the ontology level first, only showing potential exploration paths in the datasets. However, the approach should allow users to quickly access the individual instance data if required.

⁴A *multi-pivot* approach is defined as one where instead of using a single trail of navigation, the interface allows multiple trails of navigation to explore a dataset, independent if this is on the instance or ontology level. Thus the rest of design decisions described are strictly related to the specific instance of an interface using the multi-pivot approach.

R3. Allow navigation to be two-directional. In addition to being able to start from multiple points, the approach needs to support navigation in both directions using both outgoing and incoming links. This capability should also be available in formulating queries to retrieve instance data needs.

R4. Query and retrieve instance data based on exploration path. Once users have created and explored a sub-domain of classes of the dataset they can query the sub-domain and retrieve instance data.

4.3 User Interface

In order to test the multi-pivot approach outlined in Section 4.2 we designed a demonstrator tool called *Visor*⁵ that implemented these design requirements. Visor is a generic data explorer that can be used to access Linked Data on any arbitrary SPARQL endpoint. For the purposes of testing and evaluating Visor, we made an initial deployment on the DBPedia⁶ SPARQL endpoint. The following sections describe the user experience in Visor. Throughout the section we refer to specific areas where the description of the UI meets the design characteristics outlined in Section 4.2.

4.3.1 Data and Ontology Exploration

In Visor, exploration starts by selecting ontological classes of interest (named *collections* in Visor to provide a user-friendly name). Users can choose among the collections either by viewing an entire list of all the known collections or browse in the hierarchical view of the collections. Collections are listed in a panel on the right hand side of the user interface (Figure 4.3c). Alternatively a search bar is provided where the user can execute a keyword search to get results to both individual instances and collections of data.

Instead of choosing a single collection as a starting point of exploration, Visor allows users to select multiple collections simultaneously. The UI represents a canvas where a graph rendering of selected collections takes place. The graph rendering consists of the following nodes:

- **Selected collections.** Collections selected through the collections menu or searched are rendered with the title of the collection on top (Figure 4.3a).
- **Relations.** If there are properties linking instances between two selected collections, the interface indicates to the user that items from these collections are inter-related by displaying an arc with a blue node in the middle (Figure 4.3b).

⁵A demo version of Visor is available online at <http://visor.psi.enacting.org/>

⁶<http://dbpedia.org/sparql>

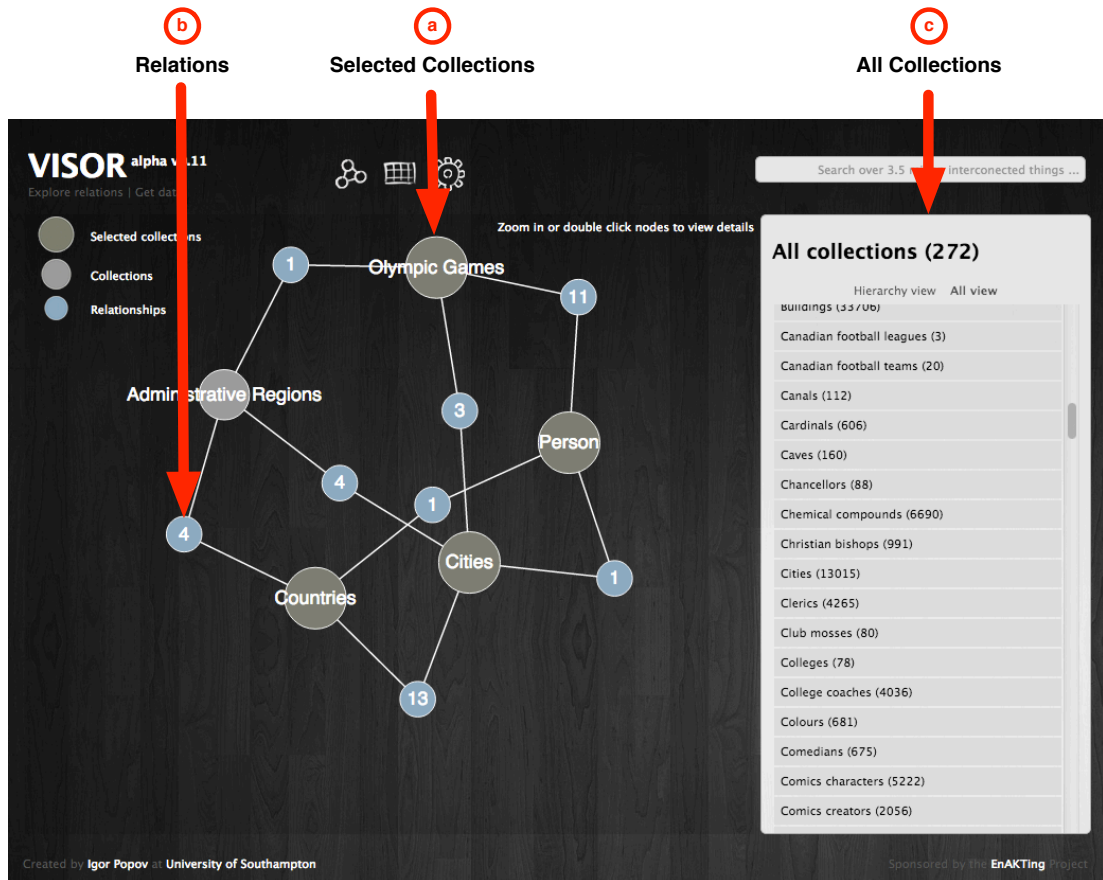


Figure 4.3: Generating a subset of the DBpedia ontology generated by selecting concepts in the ontology in Visor. Selected concepts (e.g. “Olympic Games”) are coloured in green, while suggested concepts are coloured in grey. The arcs between the two collections with a blue node in the middle indicate links between items (e.g. “Olympic Games has direct relationship with “Cities).

The number in the middle of the blue node is an indicator of the total number of properties (named *relations* in Visor) that link instances between the two collections in either direction. Since large graph datasets can have a large number of properties connecting instances of two classes, we adopt this approach to mitigate the generation of a large and incomprehensible graph (m.c. schraefel and Karger (2006)).

- **Intermediary collections.** In some cases there are no properties linking two collections. In such a circumstance, Visor tries to find the shortest path in the ontology by seeking an intermediate collection to which both selected collections can be linked. If there is none, a path with two intermediary collections is looked up. The process is repeated until a path is found. Currently, Visor finds the first shortest path it can find and suggests it to the user by adding it to the current graph. While multiple shortest paths might exist, Visor recommends only the first one it finds. In cases where users want to find another path in the ontology they

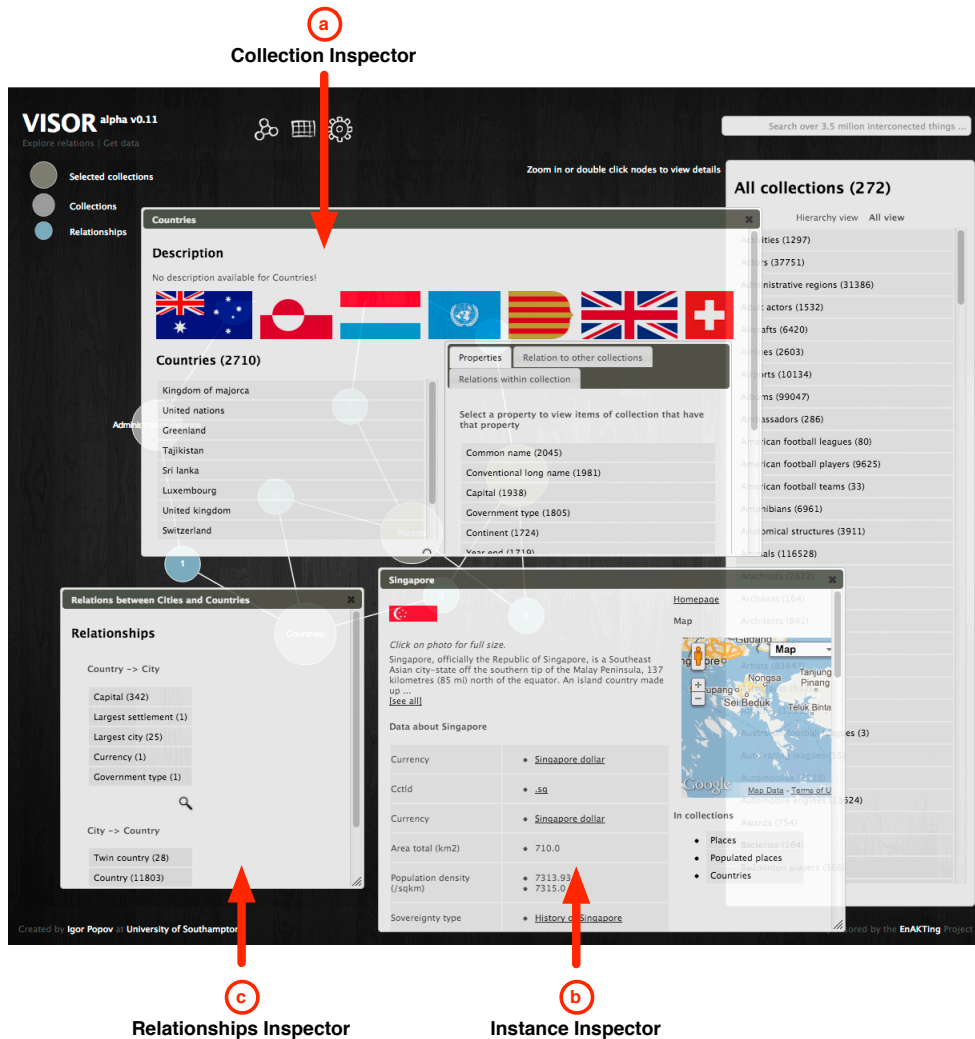


Figure 4.4: Inspectors showing diverse information on the concepts in the ontology. Information about “Countries” is shown in (a), and information about a particular country (in this case Singapore) is shown in (b). The various links that exist between “Countries” and “Cities” is shown in (c).

can simply select another collection, and the interface will attempt to link the last selected collection to all other selected collections. In this way we ensure that whatever collections are selected the resulting sub-ontology is always connected and thus query-able. To distinguish selected and intermediary collections the latter are coloured in grey and are smaller in size.

The graph representation is rendered using a force directed layout [Fruchterman and Reingold \(1991\)](#) and can be zoomed and dragged to improve visibility. Each node can be double clicked which then opens up different inspector windows. These allow a view of the details about the sub-domain. In the following we describe the different kind of inspector windows in Visor.

Select relationship to view items from inter-related collections

Connections to other collections (87)

Relation	Add collection
Leader name (To Legislatures collection) (1)	+
Currency (To Mammals collection) (1)	+
Currency (To Eukaryotes collection) (4)	+
Leader name (To Politicians collection) (165)	+
Capital (To World heritage sites collection) (21)	+
Largest settlement (To World heritage sites collection) (1)	
Capital (To Cities collection) (342)	
Largest settlement (To Cities collection) (1)	
Largest city (To Cities collection) (25)	+

Connections from other collections (1095)

Relation	Add collection
Leader (From Legislatures collection) (1)	
Meeting building (From Legislatures collection) (123)	+
Meeting city (From Legislatures collection) (123)	
Death place (From Bands collection) (23)	
Occupation (From Bands collection) (1)	
Record label (From Bands collection) (37)	+
Home town (From Bands collection) (14933)	
Genre (From Bands collection) (9)	
Birth place (From Bands collection) (108)	
Birth place (From Gaelic games players collection) (10)	+

Figure 4.5: Displaying properties in the collection inspector that link from or to items of a collection.

4.3.1.1 Collection inspector

Double clicking on a collection node brings up a *collection inspector* (Figure 4.4a). The inspector shows the individual instances which are part of the collection, a description of the collection, if available, and a list showing the possible properties that instances from that collection can have. In Visor, object and datatype properties are listed separately. Object properties (or *relations*) are shown together with a corresponding collection to which they link (Figure 4.5). Furthermore, object properties linking to and from another collection are shown in separate lists (Figure 4.5). Users can then add these classes to the canvas. In such a way we support two-directional set-oriented navigation; however in Visor we do so on the ontology level which serves as a potential roadmap for querying (Design requirement R3). Users can also view filters of instances in the inspector by selecting any property (object or datatype). This shows the instances that only have that property and show the corresponding property value.

4.3.1.2 Instance inspector

Clicking on any of the instances in the collection inspector opens up an *instance inspector* where all the data pertaining to the individual RDF resource is shown (Figure 4.4c). A simple lens that searches for commonly used properties is used to render the individual resources, including rendering images, if any exist, a description of the resource and the data associated with that resource. If geographic coordinates are found for the instance, the user is presented with a map. Additionally, we show the collections that include the particular instance (i.e. the classes that are stated as a type property in the resource). In the data panel, links to other resources open the instance inspector associated with that resource. In such a way, browsing from one instance resource to another is also supported in Visor.

4.3.1.3 Relations inspector.

The relation nodes (the blue nodes in the visualisation) can also be inspected in order to quickly access properties that link instances from two collections (Figure 4.4b). Clicking on any of the relations will display the instances from both collections linked with that property.

By selecting collections, users can create a subset of the ontology that is composed of concepts of their interest without restricting them to selecting a single collection, and use navigation (Design requirement R1) to find related collections. With the inspector windows, users can surface up the data on demand (Design requirement R2) to explore how collections are related, what are their individual instances, and if required, inspect the instances themselves.

4.3.2 Spreadsheet Creation

The implementation of the fourth requirement (Design requirement R4) is tool-specific, and it relates to how instance data is retrieved and visualised to the end user. Based on the assumption made in Chapter 3, that our first type of end-users are spreadsheet-friendly users, we designed the retrieval of instance data and queries similar to that of GEORDi. In a similar fashion, in Visor, a user generates a custom spreadsheet by specifying the parts of the explored sub-domain using the previous steps. The tool leaves more advanced representation and visualisation approaches of a more suitable and powerful tool. Thus, spreadsheets created in Visor can be exported in a variety of formats which can be picked up and reused in different applications. For example, they can be published as visualisations using ManyEyes or published on the Web as a standalone dataset using an Exhibit or simply be used in a spreadsheet application for further analysis (Viegas et al. (2007); Huynh et al. (2007b)).

In Visor, once a subset of the ontology is selected the user can query this information space by creating custom spreadsheets based on the selected concepts and relations from the ontology subset. After the spreadsheet has been created users can export their custom made data collection in a format of their particular needs. Visor currently supports exporting data in CSV and JSON formats; however the system is extendable and multiple formats may be supported. In the following section, we describe the query interface and procedure for creating custom spreadsheets.

4.3.2.1 Main collection.

The “Create a table button located in the top menu of the UI opens up a query interface which guides users in selecting things from the previously explored domain (Figure 4.6).

The first step in creating a spreadsheet is selecting the *main collection* i.e. the collection that will be the focus of the spreadsheet (Figure 4.6a). This will instantiate a spreadsheet with a single column (the *main* column) composed of the instances from the main collection. All subsequent columns added to the table will be facets of the first column, each created by specifying a path showing how the items of the newly created column are related to the items from the first column.

4.3.2.2 Adding columns.

Once the main collection is selected, adding additional columns is the next step. The first choice of columns are the datatype properties of the main collection shown in Figure 4.6b. Users can select a property and click on the “Add column button to add the column to the table. By default, when a column is added, Visor queries and tries to find a corresponding value for all the instances in the main collection. If such a value does not exist a “No value cell informs users that the item in the main column does not have that property. The default option corresponds to generating a SPARQL query with a `OPTIONAL` statement. To filter for non-empty values users can check the “Show only option before adding the column. In such a way, users have the flexibility of selecting which columns are optional and which required having a value in each cell. Additionally, users can also choose the “Count option to count the values in a cell in the corresponding to the item in the main column. Similarly, selecting the “Count option corresponds to having a `COUNT` query in the SPARQL query.

4.3.2.3 Defining column paths.

Users can also add columns based on other collections in the current sub-domain. The query interface allows users to specify a path that connects items from the main collection, to items in the newly added column. This can be implemented in two ways:

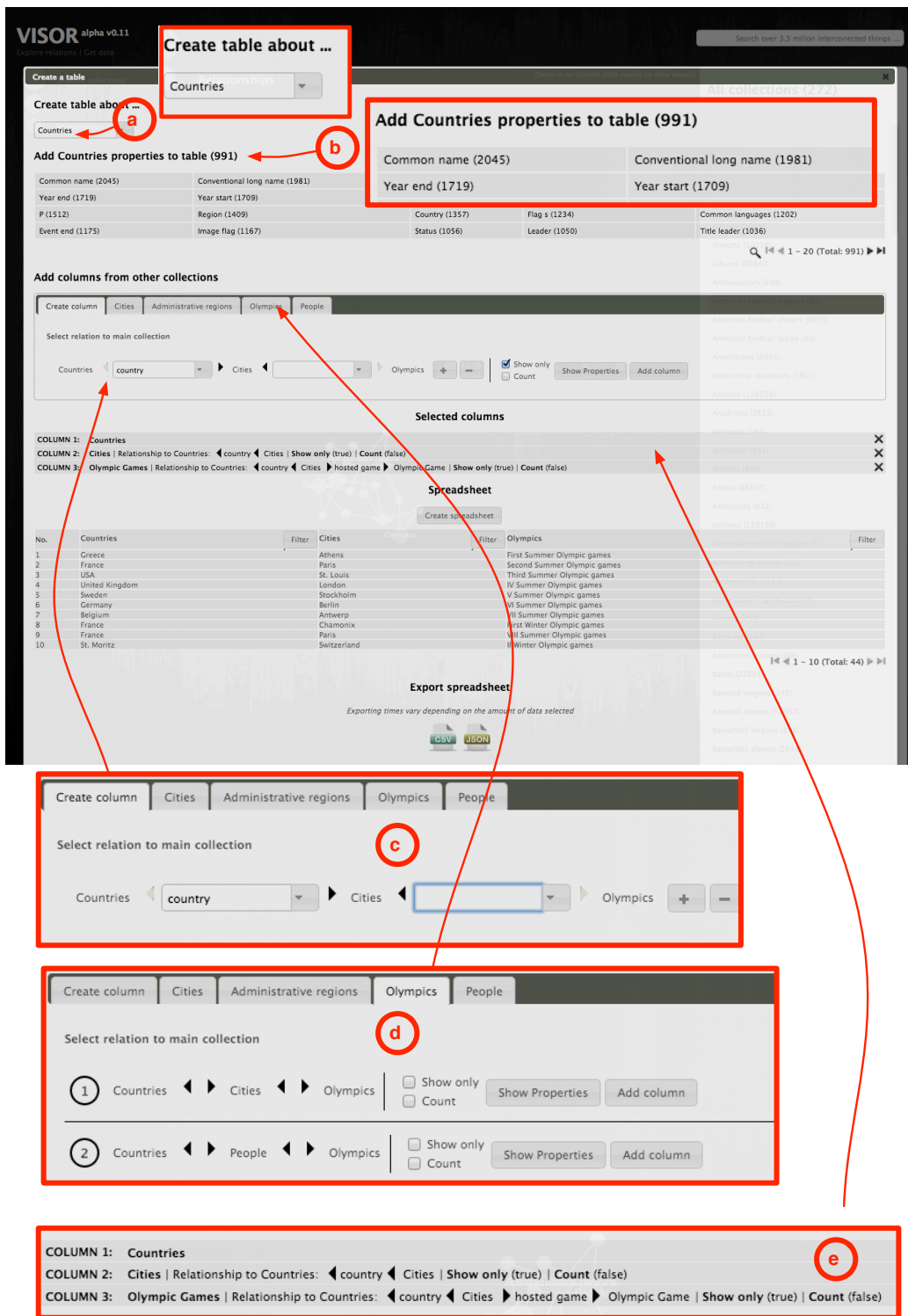


Figure 4.6: Spreadsheet creating/query interface in Visor. Users start by selecting the main collection (a) datatype properties (b) columns from other collections in the sub-ontology by specifying relations to the main collection (c) and (d). A preview of the columns is shown in (e).

1. The first way of doing this is by using a path creation tool (Figure 4.6c). The path creation tool starts a path with the first element being the main collection.

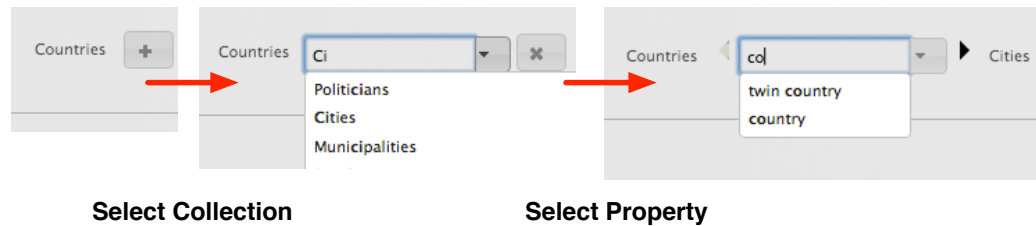


Figure 4.7: Specifying a path from the main collection to an arbitrary collection in Visor.

Users can then select a collection that is related to the main one using a drop-down list of available choices (Figure 4.7). Once a related collection is selected, a property that links them is selected again from a selection of choices in a drop-down list. Then another collection can be chained to the previous one and again a property between them is specified and so on. When a column is added based on the specified path the column pertains to instances from the last collection in the path. We note that the path creation tool enables users to connect the collections by properties going in both directions (the left and right arrows shown in Figure 4.6c,d).

2. To help speed up the process, an alternative way of adding columns is supported. In the “Add columns from other collections section of the generator interface, a tab panel allows choosing a collection that is part of the sub-ontology domain. Each tab panel contains suggested paths for reaching that node (Figure 4.6d). It lists all the paths from the main collection to the collection specified in the tab. Users need only to specify the properties in-between the collections. This saves time to the users, as well as gives cues into all the different ways that items from two collections can be related.

Users can update the current spreadsheet to monitor their progress at any time. An overview of selected columns is shown to the user (Figure 4.6e) which allows backtracking on choices made as well as rearranging the ordering. The spreadsheet also supports filtering for specific values in a column. Once users are satisfied with their custom spreadsheet they can choose to export it in a number of different formats.

4.3.3 Implementation

The implementation in Visor is composed of a front end (UI) and a back-end system. The UI is based on HTML5 and Javascript and the visualisation of the ontology was implemented using Protovis visualisation toolkit (Bostock and Heer (2009)). The Visor back-end server is a Python/Django application that serves data in a JSON format to the front-end by exposing a RESTful interface. Thus the UI side of the application does

not rely on any raw SPARQL query generation or parsing SPARQL results. The system is generic, and can be used on any Linked dataset that is available through a SPARQL endpoint.

4.4 User Study

In order to ascertain whether people will be able to learn and use Visor to explore unfamiliar datasets we conducted a user study. The purpose of our study was twofold: (1) we wanted to test if there were any major issues in the users ability to comprehend and use the tool and (2) identify specific usability problems and areas where interaction can be improved. Thus, the goal of the study was to test if our approach was viable.

4.4.1 Study Design and Procedure

For the study, we recruited ten participants through an email advertisement among graduate students at the University of Southampton. Seven of the participants were male and three female and their ages ranged 21-41. All users were regular users of spreadsheet tools. Additionally, we wanted to have a diverse group of users with respect to gained knowledge of Semantic Web/Linked Data technologies and see if there were any particular difficulties among users with different skill levels. We asked them to rate their knowledge of Semantic Web/Linked Data technologies on a scale of one to three, one being *"very basic understanding or no knowledge"*, two being *"some knowledge and understanding"*, and three being *"high or expert knowledge"*. To gain further insight in their skills, we asked them to rate their knowledge on the same scale to several specific areas: (1) Linked Data application development, (2) Use of SPARQL, and (3) Ontology Engineering and/or data authoring. Half of the participants had no or very little understanding of Semantic/Linked data technologies, 3 participants had intermediary knowledge and 2 participants had expert knowledge. No users had any prior knowledge about the structure of the DBpedia dataset used in the study.

The study used the cooperative protocol analysis or "think aloud method (Dix et al. (2003)). We chose this method because we wanted to pinpoint any potential usability issues introduced by the design requirements R1-R4 and get users insights into identifying the problems.

Each participant went through a study session that took approximately one hour to complete. A session was structured in the following way. First, the participant was shown a 6 minute video⁷ tutorial of Visor. The tutorial explained the terminology used in the tool and showed a complete example, worked out in Visor. Second, the participants were handed three written tasks to complete. During this time the "think aloud protocol

⁷The video is available at: <http://vimeo.com/24174055>

was observed, and we recorded the users screen and audio. Finally, participants were required to fill in a questionnaire, in order to reflect and give feedback based on the entire session with questions targeting specific portions of the UI.

Two of the tasks were structured tasks i.e. the users were given a concrete task with a clear result. The first task required a three column table with generated one-hop links in a single direction, while the second required more columns, specifying a loop pattern, and setting paths with two-directional patterns. The third task was unstructured i.e. we gave users a general area of browsing and exploring and coming up with some data of their particular interest. The following tasks were used in the study:

- Find all the parks located in cities, show the city and the country where the city is located
- Find royals who have intermarried and find the country they came from
- Find data of your choice related to universities, scientists and science awards

4.4.2 Results

During the execution of these tasks, we focused on observing three things: (1) Observing user actions during data finding tasks, (2) observing when users chose to view the actual instance data and for what reasons and (3) observing what problems participants experienced when attempting to create their spreadsheets.

Data finding. When searching for collections to build their sub-domain in order to complete their tasks, most participants (nine out of ten) chose to use multiple collection selection rather than navigation after selecting their first collection. Only when the resulting connections contained intermediary nodes that did not meet the requirements of the sub-domain, did they resort to navigating through other potentially useful collections. This was particularly the case during the exploratory task. Most users used a keyword search option to search for collections. Beyond using it for finding collections, participants suggested additional ways that search can be useful for finding additional data. For example, one user commented that the use of synonyms would be helpful to find collections. Another user, for example, wanted to search for a particular instance during the exploratory task because the user wasn't sure in which collection that particular item can be found. Beyond searching for instances and collections, we observed that some users tried looking up things that we currently did not support e.g. searching for relations. For example, one user thought that there might be a collection named "Spouses before realising that it might be a relation instead.

Showing instance data. We also observed users in order to see how much they would need to rely on viewing the underlying instance data during various stages and across

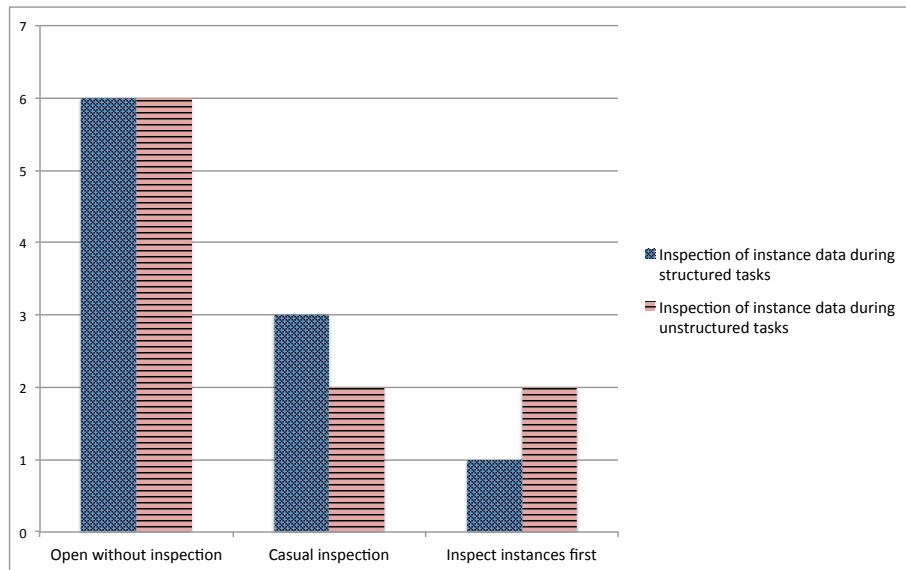


Figure 4.8: Reliance in showing instance level data as opposed to collection overviews in Visor.

different tasks. We concentrated our observations on two things. One was to observe if users needed to examine the underlying data during the initial exploration phase when the user was building a sub-domain of data. Second, we wanted to test if they can specify relationships that were three or more hops away without seeing the intermediary related data. During the structured tasks, we found that users spent very little time or none at all exploring the generated sub-domains with the inspector tools (Figure 4.8). Six participants chose to directly open the spreadsheet creation tool mentioning that they felt confident they had everything they needed to answer the query. Three others noted that they just wanted to open up the inspectors to explore the instance data before making a query, however, they mentioned no particular reason except a casual exploration. While we observed a slight increase in using the inspector tools during the exploratory tasks, we did notice that the spreadsheet creation tool was used as an exploratory tool as well. When creating their spreadsheet more than half of the participants chose to view their progress with each added column. At the start of the sessions, novice and intermediary users reported difficulties in grasping how paths worked, but once they were explained they felt confident in generating paths two or more hops away without viewing the intermediary data.

Spreadsheet creation. Users found the spreadsheet creation tool was the most difficult part of the interface the user can learn and use. Some suggested better integration with the visualisation by either selecting or being able to drag and drop directly from the graph into the header row of the spreadsheet. In hindsight, we realised that a simpler design, one that is directly integrated with the visualisations in the system would have produced a much more usable design. Designing a simpler spreadsheet creation tool in this case is purely a matter of refining the system as part of perfecting a prototype tool.

While for the two expert participants and one non-expert specifying the direction of relationship made sense, for most other participants specifying the direction of relationship seemed irrelevant. When asked what they would prefer, most of them responded that they would like a single list of how to relate two collections instead of two separate lists. However, we observed that users had no difficulty in specifying paths when two-directional patterns occurred.

When faced with the choice of using the template paths or create them manually, the preference of participants was split among these two choices. The participants recommended, however, that they would like to be able to reuse paths from existing columns which were sub-paths of the new path, rather than specifying a new path every time they add a new one.

Task completion and survey. Task completion was generally high: eight out of ten participants were able to complete all three tasks and create spreadsheets to the specified requirements of each task without any errors. Overall, we found that the users were able to easily learn and create their spreadsheets after the one-hour session. After going through the tasks, participants were asked to submit a survey and rate the overall difficulty of using the tool on a Likert scale of one to five. Two participants reported that they found the tool very easy: (1) to use, six reported it easy (2), one user reported it average (3) and one user reported it difficult (4) to use. When asked to rate specific tool components or functions, most participants (8 out of 10) reported that the graph visualisation is useful and easy to use, while they gave the spreadsheet creation tool an average (3.2) score.

4.4.3 Implications for Design

Based on the results we can compile a set of recommendations which could be considered useful to future designers of data-centred exploration interfaces.

Integrate keyword search with direct manipulation techniques. The multi-pivot approach showed that a more flexible approach to exploring data is a viable approach for exploring Linked Datasets. We noticed that users not only took advantage of this flexibility, they even wanted more freedom when trying to find the portion of data domain of their interest. Thus rather than being able to just add multiple collections, users should be able to search more freely for things, such as properties, instance data and view how they are relevant in the already explored data. Currently Visor only allows adding collections - a future tool that allows discovery and structure understanding, such as Visor, can make all of these objects first-class citizens to the visualisation. Integrating keyword search techniques that allow finding these with direct manipulation techniques that support the discovery and exploration process may further improve exploration of unfamiliar datasets. So far, the two techniques have been deployed separately; direct manipulation

techniques have been mostly focused on supporting direct manipulation techniques for navigating graph data only, while interfaces supporting keyword search have focused on entity retrieval or answering questions. Future designers of data browsers should be encouraged to consider closely integrating keyword search with direct manipulation features of the data browser.

Support two-directional navigation. In Chapter 3, we noted that some interfaces support navigation in a single direction only i.e. only from outgoing properties. This limits the expressivity of the queries that can be answered by the data browsing tool. While from an implementation point, supporting back-links on the open Web of Data is much harder a task than when the data is contained in a single store, our study shows that from an interaction point of view supporting both does not have any significant impact on users when browsing. Thus back-links and forward-links can be treated equally.

Show data on demand. One of the things our study shows is that users can browse and query for data without relying on viewing instance level data. As we pointed out in Chapter 3, most browsers visualise exploration of a dataset by representing instance level data in their tools. However, when exploring unfamiliar datasets showing instance level information does very little to convey structure (i.e. schema) level information about a dataset. Instead of real-estate being used on massive amount of instance data, overviews and other summarising information about the dataset can be used to improve the understanding of unfamiliar datasets before querying and retrieving instance data. Retaining context while exploring or combining querying with visual aids can be utilised to give an overview of the exploration path and make querying easier. Users should, however, always retain the option of viewing the instance data of the current result of an exploration at any time. Visor is one example, using a graph-based visualisation to summarise ontology level information and give access to instance data on demand. Other types of visualisations might be used. For example, [Zhang and Hefin \(2011\)](#) use a tag cloud representation of a dataset to summarise the content and provide statistics about a dataset.

4.5 Summary

In this Chapter, we examined some of the interaction challenges associated with data navigation as an exploration technique in data browsers. To address these challenges we presented multi-pivoting, an extension to pivoting that aims at improving exploration of unfamiliar datasets. We presented Visor, a tool that implements this approach by allowing users to create their own sub-domains of interest by combining selection and automatic link recommendations. Our study showed that users were able to find and solve tasks from a large dataset that was unfamiliar to them. Our approach shows

that with proper tools even large Linked Datasets with complex domains can be effectively explored. Future work might produce different approaches that extend or improve existing approaches for exploring unfamiliar domains.

Chapter 5

mashpoint - Browsing the Web along Structured Lines

The work presented thus far focused on generic data browsers - tools that allow access to Linked Data on-demand. In most cases, generic data browsers make raw Linked Data available to non-technical users. In Chapter 3, we proposed two personas and thus described two scenarios, in order to distinguish use cases where users engaged in data-centric interactions needed access to raw data, from users that engage in data-centric interactions through data-driven applications. So far, the tools we presented focused on the first type of users. In this Chapter, we tackle the problem of providing data-centric interactions for casual users, users that experience data only through data-driven applications.

As our analysis of data browsers in Chapter 3 showed, all generic browsers exhibit similar attributes: they use generic data representations, use limited heuristics to figure out how to represent the data, and allow navigation through graph data (usually using pivoting i.e. set-oriented navigation). Additionally widgets and tools can be provided in order for the users to quickly create domain dependent data representations upon finding data of their interest. However, navigating and finding data in generic browsers, even when abstracted from its machine readable format is still not a trivial task: it requires finding the right data from potentially large data sources, figuring out which data would be interesting or need to be combined, gathering the data and finally representing them in a format other than the one supported by the generic browser. This might be too much of a price for users that are used to just point-and-click browsing and keyword search engines to find information on the Web. While in the future, recommendations based on additional representation knowledge might improve the usability of generic browsers, we are a long way off having generic browsers accommodating automatic representations and recommendations for all possible Linked Data sources on the Web. On the other hand, in Chapter 2 we examined data-mapping approaches, approaches that allow developers

to quickly assemble powerful visualisations and data-centric interactions over data using components, libraries or custom solutions. These tailor-made interfaces, although useful and usable, are designed to be grafted over limited datasets and thus do not enable integrating information from a Web of Data. Thus, a trade-off is evident when we are confronted with choosing among generic, data-mapping and custom-made approaches: either we need to sacrifice navigability and the affordances of using Linked Data as a resource that allows easy data integration and be satisfied with islands of applications over limited data sources, or use generic approaches that allow access to raw data, making it unusable for large numbers of users that experience data only through rich data-driven applications.

In this Chapter we present mashpoint - a framework that attempts to find middle ground between these approaches. The framework allows data-centric applications to be linked based on the similarity of entities in their datasets and thus be used as a higher level abstraction or lenses over linked data on the Web. Linked in this way, data-centric applications can pivot with selections of entities, enabling similar data-centric navigation found in generic browsers to be performed using distributed data-driven applications.

This Chapter is organised as follows: first we present the overall concept and design of the mashpoint framework, and discuss how mashpoint can be looked at from different perspectives of data browsing. We then discuss the case of having data-centric applications as higher level abstractions over the Web of Data and how this approach can be used to effectively solve a number of challenges common to generic data browsing. Next, we briefly present implementation details of the mashpoint prototype. We then report on an exploratory user study we carried out to identify any usability issues with using applications linked with mashpoint to solve data-centric tasks. Based on the study, we introduce improvements to the framework and carry out a second, more comprehensive study to ascertain the effectiveness of the solutions to the issues identified in our first study.

5.1 mashpoint - Concepts and Design

The Web today is populated by many applications offering data-centric interactions. In Chapter 2, we noted how data-centric features are a pervasive feature on many online sites offering content based on data. Such online websites exhibit similar patterns of design when offering data-centric features. They usually provide filtering and finding data either by categorising or through faceted browsing interfaces over single or multiple collections of common items. For example, an online shopping site might provide and filter information about various products. A travel website might be able to filter through hotels and flights. While different websites might offer different information about what

are essentially the same entities, it is difficult to easily combine data from multiple sites or filter one sites information by another sites filters.

To better illustrate this point, we again consider the example described in our second scenario in Chapter 3. The scenario describes a situation where a user, in this case Anna, needs to combine and filter information about hotel bookings. In the scenario, Anna needs to find the hotel and flight bookings on travel websites, but filter only for hotels which are on the conference discount list and further filter those that have good transport links to the conference site. Let us imagine that Anna needs to combine this information from three websites: first a travel website containing information about flight and hotel packages; second, the conference website which lists information about the discount hotels, and a local transport website that provides transport links though inputting location information e.g. postcodes. As we described in the scenario, currently Anna would not be able to easily filter the information found across these three websites; she would have to go one hotel at a time, copying and pasting information between these websites and checking if the information is about the right hotel and if it meets the criteria she has set for finding the right hotel. Since she is looking up multiple hotels, her task would be time-consuming. As we can see these websites have various information about the same entities; in the case of hotels both the travel website and conference website have information about “Hotel entities, while the conference website and the transport website have various information about “postcode entities, and in the case of the conference one, the postcodes of the hotels, while the travel website links the postcodes it recognizes with local transport information.

On the Web, applications that are built either by deploying the underlying data as Linked Data or through reusing Linked Data will expose URIs to denote individual entities in the data. While different publishers might provide different URIs, publishing practices on the Web of Data requires users to link to other similar resources in order to identify semantic similarity. Thus, if entities between applications are reconciled, a simple communication protocol that allows entities to be communicated from one application to another can be used to allow applications to cooperate and answer data-oriented needs. With the mashpoint framework, we leverage the fact that data-driven applications built on data from the Web of Data will be able to expose entities that are uniquely identified, and we use this fact to provide a framework that allows for pivoting with data between applications. In the following section, we describe the pivoting between applications in terms of the users experience.

5.1.1 User Experience

To illustrate pivoting between applications we present an example of two applications that have already been linked by using the mashpoint framework. Figure 5.1 shows two different applications that are connected through mashpoint. The first application

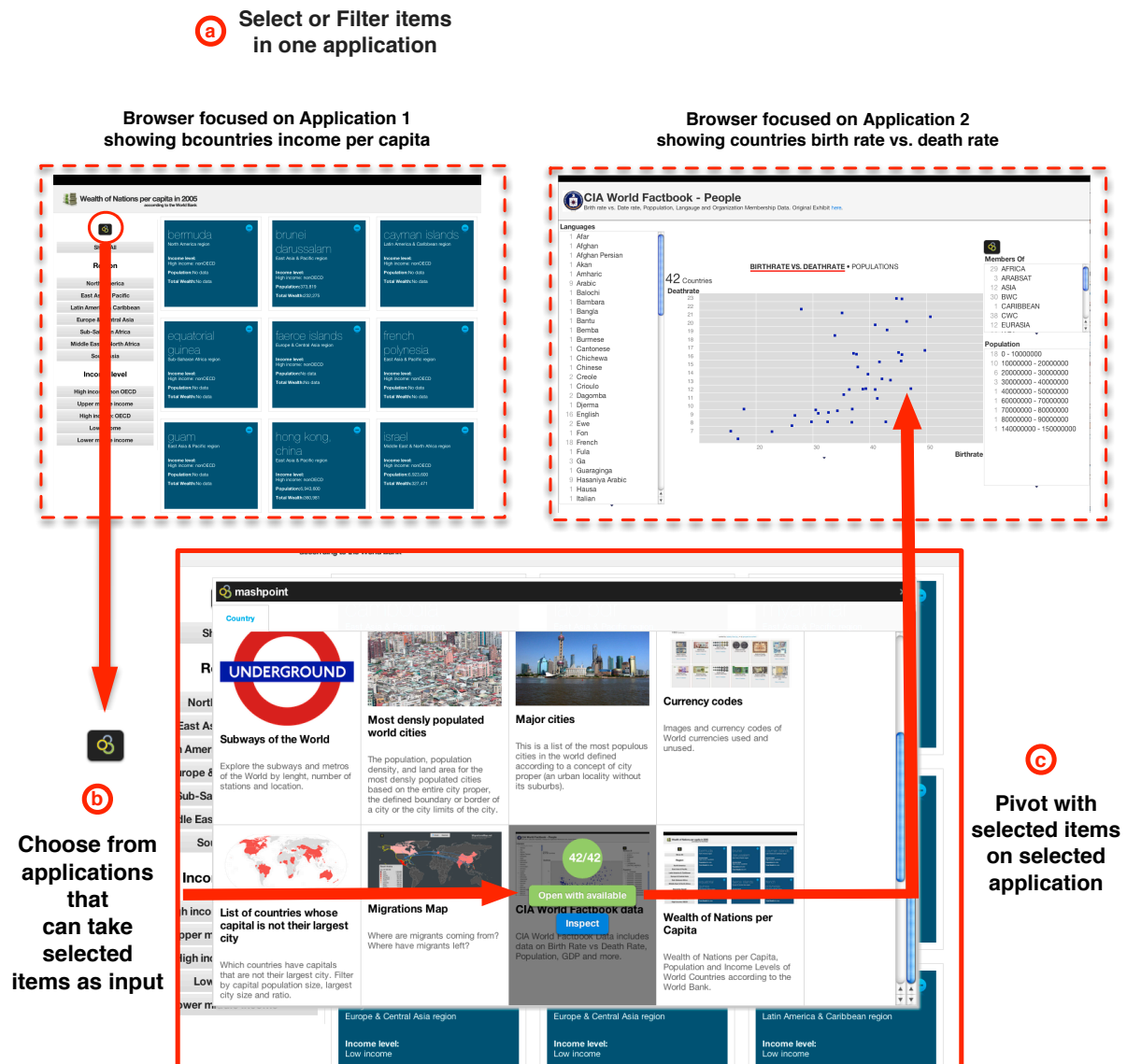


Figure 5.1: An example of pivoting with data between two applications in mashpoint.

(Figure 5.1a), is a simple data-oriented application showing levels of income per capita for world countries. It allows users to view population and income levels of countries, as well as to filter them by a number of facets: geographic area, income level, membership in international organisations etc. The other application in our example, is another data-driven application that shows data about countries from the CIA Factbook¹. The application shows charts of birth-rate versus death-rate and allows this information to be filtered by countries population, region etc. As we can see, each application has different information and provides different filters over the same entities, in this case countries. Let us suppose now, that we are interested in finding the birth-rates vs. death-rates for countries with a high GDP per capita. Neither applications hold all the

¹<https://www.cia.gov/library/publications/the-world-factbook/>

data we need to answer this question; the first application has information about GDP per capita and allows filtering by income levels but has no information about birth-rates or death-rates. Conversely, the second application does have data about birth and death rates, but no information about GDP. When applications are linked in mashpoint two things are enabled: (1) applications that share similar entities can be discovered, and (2) a selection of entities in one application can be communicated in another application, thus allowing pivoting between applications that share similar entities. This interaction is shown in Figure 5.1. When added to the mashpoint framework, each application embeds a mashpoint button (Figure 5.1b), which, when clicked, opens up a window that offers other applications that can take the current selection of entities and new insights about the selected entities in the first application. In the example, we first filter the application to get countries that have a high income. When the mashpoint button is pressed, the window shows all the other applications that can show information about either all of the selected countries or a subset of them. We can then select the CIA Factbook application from the list. When shown, the list of countries in the CIA application will reflect the selection we made in the first application.

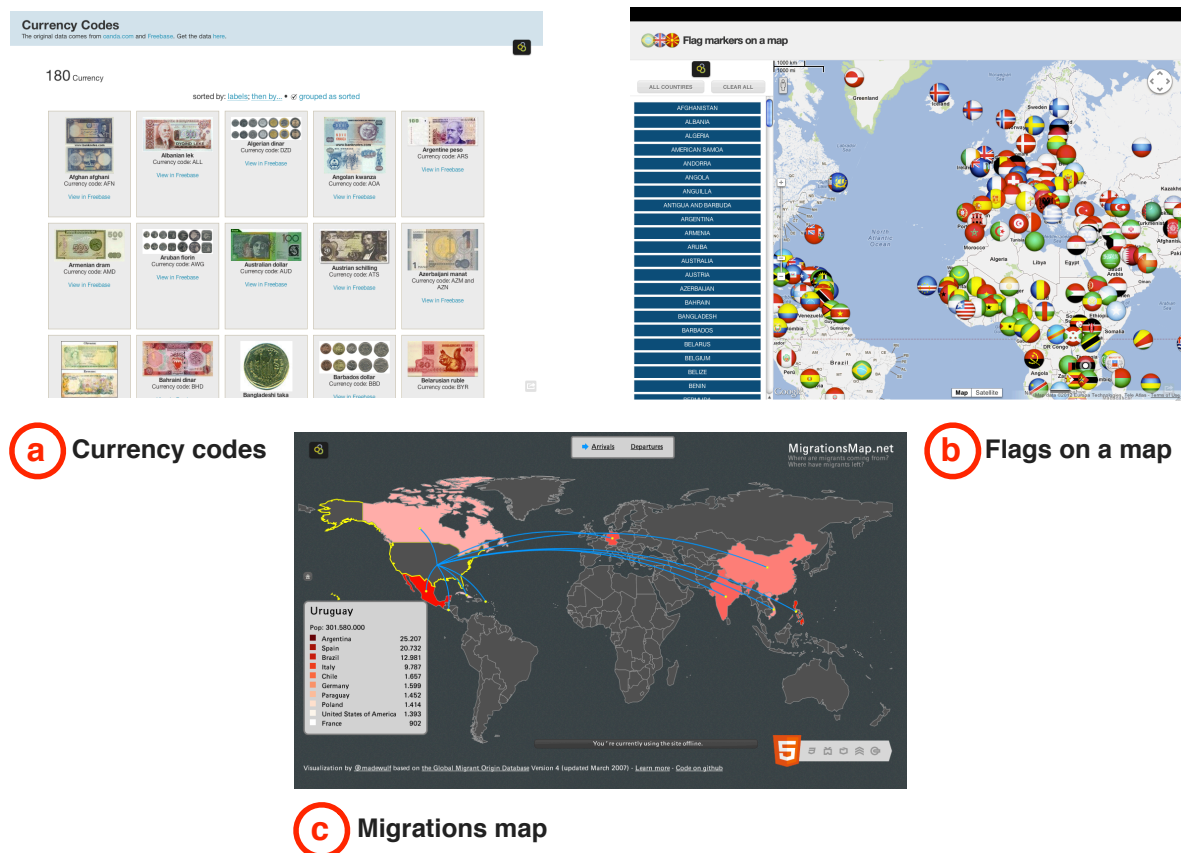


Figure 5.2: Example applications linked with the mashpoint framework.

As part of the development of this framework, we started adapting and linking existing data-centric applications on the Web. Figure 5.2, shows three other applications that

were adapted and linked up using the mashpoint framework. The first one² (Figure 5.2a) is a simple Exhibit showing images of world currencies, and the currency code. The second application³ (Figure 5.2b) is a simple exploration application which allows users to view and browse countries flags depicted on a map. The third application⁴ (Figure 5.2c) is an existing open source application found on the Web⁵ displaying migration data geographically after being integrated with the framework.

In the following section, we describe several examples how combining and navigating through different data selections can produce some interesting insights into the data:

- In the previous example we used an application that showed World Bank data about GDP/per capita (Figure 5.1a). A user can browse data in that application using the provided facets; however, the application provides only a single data representation, a list of countries and the corresponding information. A user may wish to view countries geographically on a map in order to see how countries of different income groups are distributed geographically (for example, where are the most “Low income countries located?”). Using current tools on the Web, a user would be required to copy and paste each country in another application (e.g. Google Maps) to answer this question. Using mashpoint, however, the user can take any selection of data and find applications that are able to provide geographic information and data representations about the data. For example, after filtering for “Low income countries, the user can open the mashpoint dialog and select the Flags on a map application (Figure 5.2a), which can display the current selection of countries on a map as little flag markers. It is immediately clear that out of all the low income countries only a single one (Haiti) is in the Americas, while the rest of the low income countries are in sub-Saharan Africa, Central and South-east Asia.
- Drawing from the previous example, once viewed geographically, a user can choose to view additional data about the selected “Low income countries by pivoting to the CIA Factbook application (Figure 5.1b) and explore data about birth rates and death rates for the selected, low income countries. If a user decides to compare these with high income countries, a user can repeat the same navigation, only this time starting with high income countries in the first application. A user can then conclude that there is a great diversity in both birth rates and death rates in low income countries as opposed to high income countries where death rates are fairly consistent, and birth-rates experience small variations.

²<http://mashpoint.net/demoapps/currencycodes/index.html>

³<http://mashpoint.net/demoapps/flagsonamap/index.html>

⁴<http://mashpoint.net/demoapps/mapmigrations/index.html>

⁵<http://migrationsmap.net/>

- Similar to the previous example, a user might choose to view migration patterns of countries in a particular geographic region. For example, pivoting from Middle-eastern countries selected in the Income Levels application (Figure 5.1a) to the Map Migrations app (Figure 5.2c) can reveal to the user that people from those countries typically migrate to countries in the same region and to countries of Western Europe and Northern America.
- A user is planning a trip across Europe, traveling to multiple European countries. While aware that some European countries share a single currency, the user, not knowing if all the visiting countries use the same currency decides to check this information for each country separately. Deciding that the best way to quickly select the countries of interest is to use a map, the user selects the countries of interest on the Flags on a map application (Figure 5.2a). By selecting the Currency codes application (Figure 5.2b) the user is able to pivot with the current selection of countries, obtaining the corresponding currencies of each country. This saves time to the user since the alternative would be to look up each country and integrate the information manually.

If we examine all of the above examples carefully, an interesting observation can be made of each example. Each application by itself offers very limited capabilities - they only allow data-centric interactions over the dataset used in the application. By enabling selections of entities to be pivoted or shifted to other applications, we allow users to interact and complete data-centric tasks that usually require tedious manual work.

5.1.2 Interaction Concepts

We notice that in the examples we've provided so far that all of the applications provide data around a single collection of entities i.e. countries. As we know, browsing and using Linked Data connects information and data about entities of different types e.g. countries link to cities, people etc. Additionally, we notice that all of the applications have the same cardinality i.e. all the applications have information about all the countries. This might not always be the case. In this section we describe different situations that can occur as a result of data-oriented navigation in mashpoint and we describe how these situations are handled in the framework.

5.1.2.1 Homogeneous Collections between Applications

Entities used in two different applications are considered homogeneous when both the application we are navigating from and navigating to are centred on a common information concept. In our previous examples, all applications provided information about countries. Two different situations might arise when navigating between applications

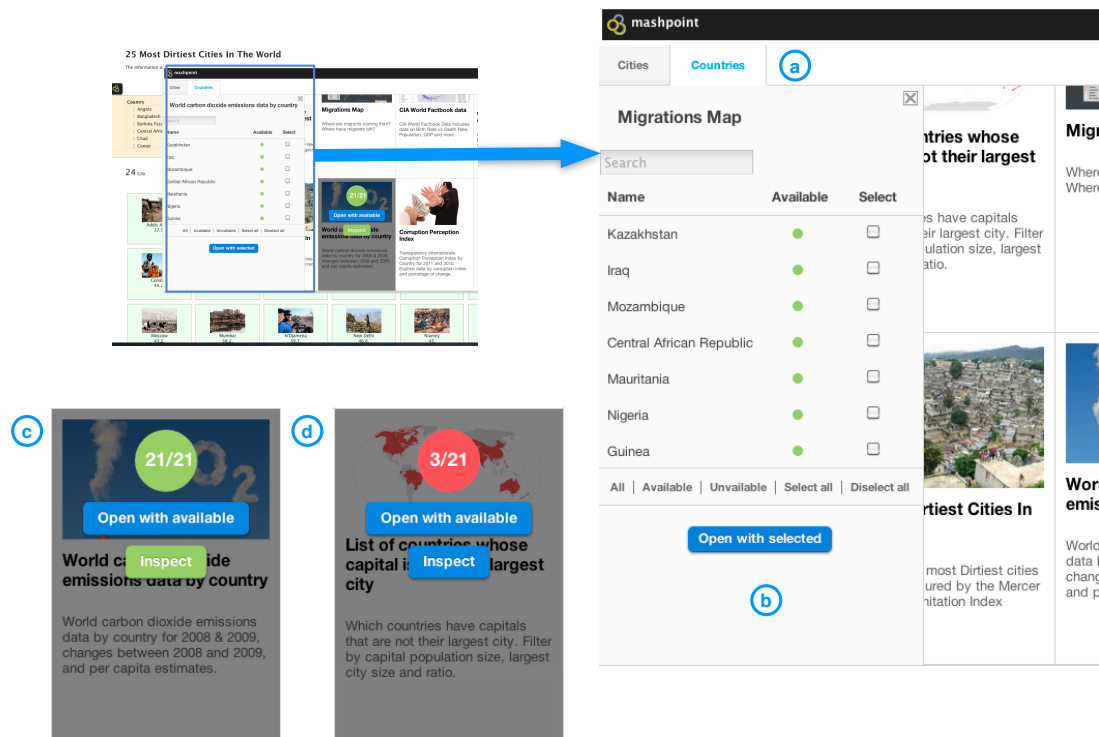


Figure 5.3: Displaying different types of information in the mashpoint application selector window.

with homogeneous collections. The first case is when the cardinality of the entities in each application is the same. For example, two applications can have an identical set of entities i.e. if they both surface entities which correspond to countries and they both provide data about all the countries in the world. This means that for every entity selected in the first application, a corresponding entity will be displayed in the navigating application. The other case is when the cardinality is not the same i.e. when the application users are navigating to gain information about a subset of the entities in the first application. For example, one application might have data about all the world countries, while another holds data about EU countries only. To solve these situations for users we added additional tools in the mashpoint window (Figure 5.3). First, whenever the user hovers over an application selection, the user is presented with information on how many of the entities will be displayed in the selected application. If an application can show the full range of selected entities, a green indicator is displayed (Figure 5.3c); otherwise the indicator is red (Figure 5.3d). There may be situations where users might want to get additional information about exactly which entities can or cannot be displayed. To do this, users can hit the *Inspect* button, which opens an *Inspector* sidebar on the left side of the mashpoint window (Figure 5.3b). The sidebar shows the list of all the entities and indicates which ones are available, and which ones are not in a selected application. Users can search or filter for particular entities. Additionally, users can select any subset of the available entities and restrict navigating to an application with only that particular selection of entities.

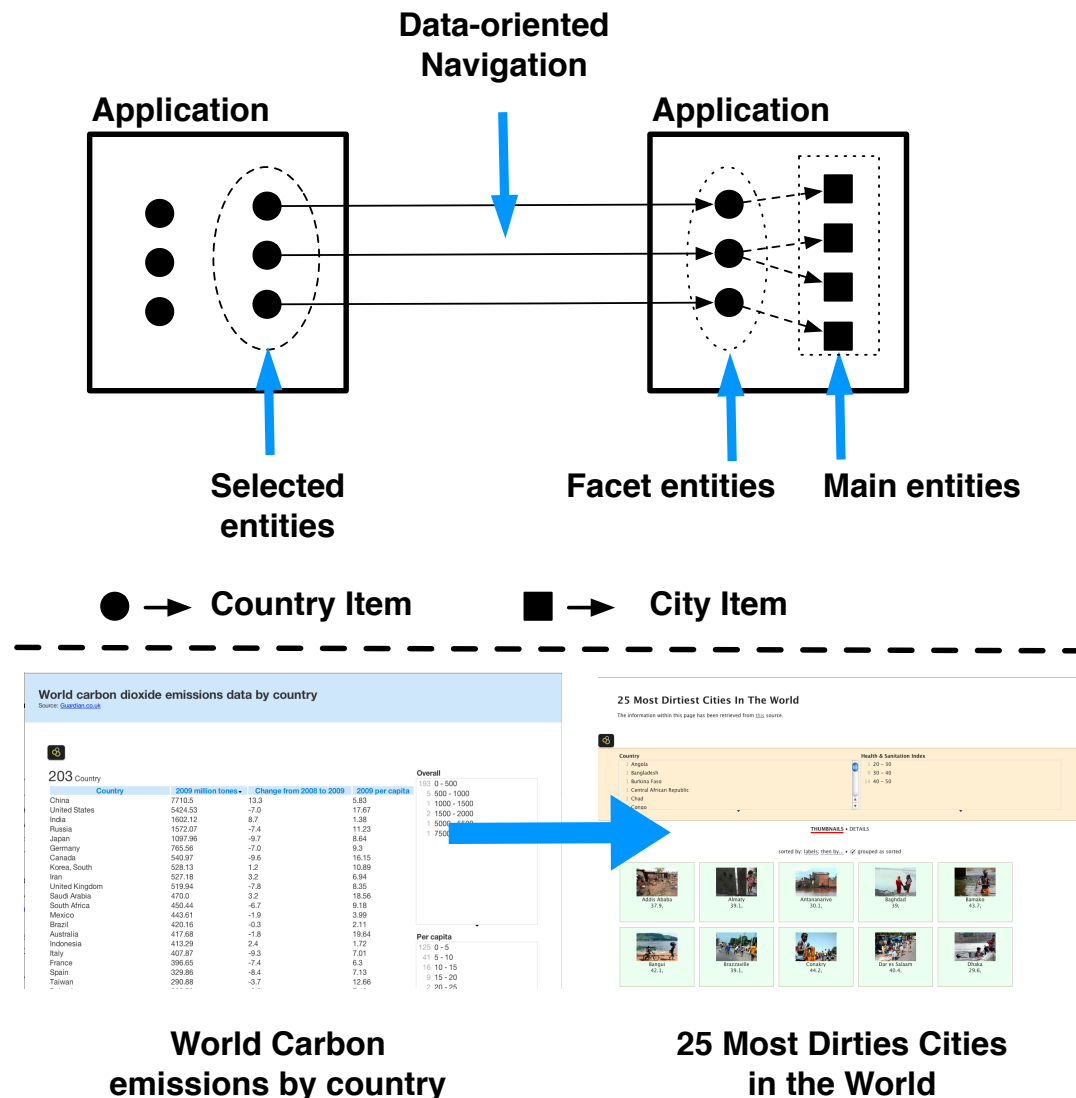


Figure 5.4: Navigation between applications which support entities from different information concept types.

5.1.2.2 Different Information-concept Collections between Applications

A data-oriented application can support entities of many different concepts. For example, an application can combine information that includes cities and countries. The application might choose to show data about cities, and include facets such as countries, to enable filtering of the cities by country. In mashpoint, both sets of entities can be exposed as separate collections of entities. Since, the application exposes country entities as well as city entities it can choose to pivot applications with either set of entities. This is shown in Figure 5.4. The first application shows data about countries CO2 emission levels, while the second application shows a list of the 25 most polluted cities, which can be filtered by countries. If we select a subset of countries in the first application, we can then see which of the selected countries have cities which rank in

the top 25 most polluted cities. In mashpoint, each collection of sets is treated separately - one can choose to pivot other applications with one or the other set. Figure 5.3a shows an example where the mashpoint window provides a way for users to choose whether to pivot applications that accept the “City entities or to applications that accept the “Country entities. Although navigation between instances of the same type corresponds to navigating through a `owl:sameAs` property in a generic browser, by enabling an application to expose several collections of entities, mashpoint provides data browsing capabilities equivalent to generic browsers navigating between RDF instances of different types, linked through an arbitrary property.

5.1.3 Defining mashpoint

By enabling applications to exchange information about entities, mashpoint enables data pivoting between multiple applications. In effect, the mashpoint framework allows the same interaction mechanism of pivoting that has been discussed throughout this thesis, but rather than doing it on the raw data level like most generic browsers, it adds a level of abstraction to the raw data by using applications built over Linked Data. However, mashpoint can also be viewed from several different perspectives. In the following section, we discuss some of these perspectives.

5.1.3.1 Lens Perspective

One way of looking at the applications connected through the mashpoint framework is to view each application as a lens over some Linked Data. Figure 5.5 illustrates this perspective. The figure shows ontology of data about countries, their population, size, birth rate, death rate, GDP, and currency including a currency code, and currency value. The circles in the figure represent classes in the ontology, while the ellipses represent literal data entities. Applications that are built over subsets of this data (depicted by the dashed-line rectangles) can be seen as lenses or views over the data. Each application can show one or more collection of items. For example, the first application (Application 1) can be seen as a lens over data about countries by taking information about the countries birth rate and death rate and representing the information as a scatter plot. The second application (Application 2), on the other hand, takes different data about countries (in this case area size, population and GDP) and provides a faceted browsing interface over countries and their GDP per capita. It holds facets such as geographic area and different income levels. The third application (Application 3), on the other hand, shows data about two collections: currencies and countries. The application shows diverse currency information, such as currency code and value, as well as the country which can be used to pivot with the other two applications that have information about countries.

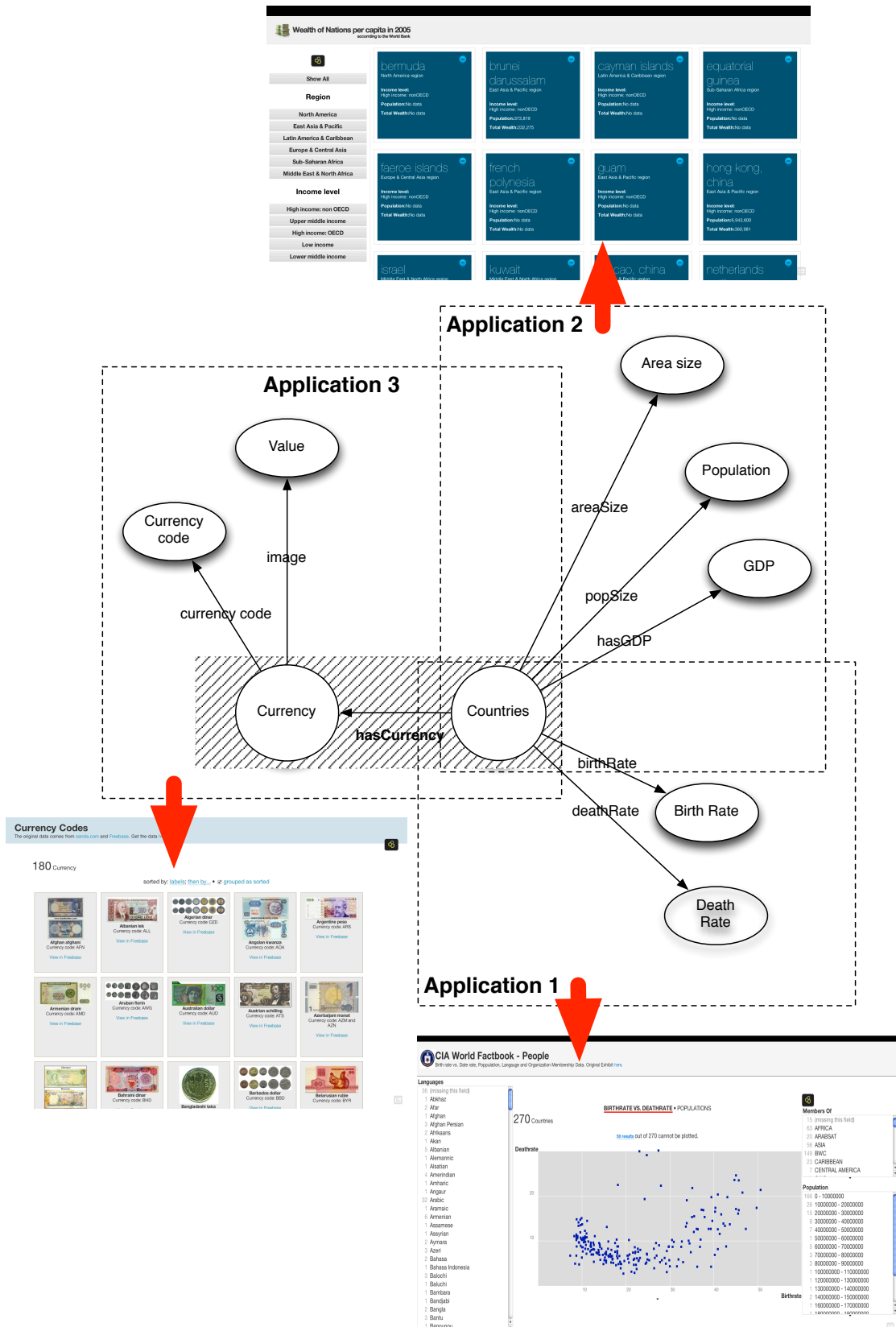


Figure 5.5: Applications in mashpoint viewed as lenses over a graph of data.

5.1.3.2 Application Pipelining Perspective

Another way of viewing the link between the applications is by considering them in a pipelining metaphor. In computer science the concept of pipelining is used to consider applications that can be chained - i.e. the output of one application can be used as an input in another application. This concept of interactions dates back to pipelining of Unix-based terminal commands and later on in graphical user interfaces (GUIs) by dragging and dropping between applications. Applications in mashpoint can be viewed in a similar analogy. Each application can be viewed as an application that can take some input (in this case a collection of entities) and can produce some output - produce an interface that is able to represent and further manipulate the data. Once the user manipulates this output using the tools in the second application, it can again be used as an input in a third application. The use of data across different applications processed over RDF as a same data model has been discussed by [Huynh and Karger \(2009a\)](#) although in a more generalised notion than just browsing.

5.1.3.3 Structured Data Clipboard

Yet another way to view mashpoint is to view pivoting between applications as a structured data clipboard. In one of his design notes⁶, Tim Berners-Lee suggests that one way of looking at the Semantic Web is to view it as a way of breaking down barriers between applications. Berners-Lee suggests a semantic clipboard as an example where data from one application can be copied and pasted in another application. For example, copying and pasting data from a photo sharing application on a calendar application will show chronological information about the photos. mashpoint can be seen as a lightweight version of a Semantic Web clipboard. While Berners-Lee's proposal suggests that data moves between applications, mashpoint suggests that selections of entities move between applications. The latter approach is much more easily implementable and scalable, since reconciling identifiers is much easier than reconciling data in general, which requires that applications have prior knowledge of the schema of the incoming data.

5.2 The Case for Application Level Abstractions

Throughout the previous Chapters we noted various limitations and problems with generic data browsers. The mashpoint framework provides solutions and improvements to a number of problems encountered with generic browsers. In the following section, we discuss how application level abstractions can solve several noted problems inherent to generic data browsers:

⁶<http://www.w3.org/DesignIssues/SemanticClipboard>

5.2.1 Representation Problem

In Chapter 2 we noted that, unlike the Web, where each page is carefully crafted for human consumption, a Web of RDF data is purposely devoid of any presentational content as an adherence to the principle of separation of content from presentation. Therefore, the responsibility is transferred to the browser to figure out how to represent data once fetched. As shown in the tools developed so far, generic browsers operate over raw, machine readable data and thus, by design, bear the responsibility for how the data is represented to users. Since they are generic, they often resort to generic approaches to representing data in structures, such as tables, graphs etc. We also discussed various problems of adding data about lenses in generic browsers. By having applications level abstractions over data that act as lenses over Linked Data, we can always be sure that the data is shown within a rich custom made context of the application.

5.2.2 Dealing with Fine-grained Raw Data

Often times the data that the data browser exposes is much too granular information for end users, often requiring them to do complex transformations or many selections before they can complete an information related task. For example, a simple question of viewing a visualisation of countries GDP/per capita on a map will require multiple steps to complete. Normally, a user would first have to explore the graph and find resources of “Countries that will probably be associated with properties, such as latitude, longitude, GDP and population. Then, a user would have to find and specify which properties will be used to query. Moreover, GDP per capita might not be available, so a user would have to combine the overall GDP and population data before the data can be used in, for example, a chart visualisation widget. Eventually, the user can reach the desired result; however the process can be long and error prone. As our casual end-user persona described in Chapter 2 suggest, these interactions are often too complex for end users accustomed to rich, custom made applications.

5.2.3 Data Overload

Data browsing over vast amounts of data can take significant time even if filtering features are well supported. Sometimes this search will be exploratory - users might want to triage for interesting data about particular entities. But finding which data to select, combine, and visualise from vast amounts of resources can be challenging, and even if we can quickly filter the data it still requires significant time and effort. For example, if we want to visualise countries income per capita on a map we would first need to find the countries as resources. Second we would need to check if they attributes such as latitude and longitude exist so we can plot them on a map. Then

we would need to figure out if data about GDP per capita is available. It might be the case that no data about GDP per capita is available but rather that there is data about the countries population and overall GDP. Thus we would need to combine and transform these resources in order to get the desired information. Doing this repeatedly, trying to combine different properties can be time-consuming. Thus, the applications in the mashpoint framework can be seen as recommended views over sub-domains of a dataset. If we want access to the raw data, an application could provide accessing them in a generic browser. This approach would be similar to the approach of Exhibit, which allows users to directly access the raw data straight from the user interface (Huynh et al. (2007b)).

5.2.4 Social Contribution Factor

By design, mashpoint sets forward a paradigm that has an inherent, social factor of contribution, which is similar to the social nature of publishing and linking in the original Web. The reason why applying data-mapping tools such as Exhibit (Huynh et al. (2007c)) have seen much wider acceptance than generic browsers such as Tabulator (Berners-lee et al. (2006)) is because a publisher of an Exhibit can control the look and feel of data and immediately see value in providing a rich data-centric interface over data. The original Web followed a similar pattern; a published Web site offered a custom made document and presence on the Web - linking to other web sites only improved the quality of the web site by providing convenience in finding relating information. For example, a web page about events in Southampton is by itself a useful contribution to the Web, and the publisher can increase the value of information by providing links to other pages (e.g. the Wikipedia page for Southampton or other related web pages offering events information about Southampton). However, it is important to note that even without the links the web site is useful by itself. As a publishing recommendation for data-centric applications, mashpoint acts in a very similar way. Applications can be viewed as contributions which are useful by themselves - they allow some value over the data they were initially designed for. By linking them and enabling pivoting to other data-centric applications, the original application can only increase the value of the original application. In fact, this attribute may provide incentive for publishers of data-centric applications on the Web to link their data using frameworks such as mashpoint.

5.3 Implementation

In this section, we describe the implementation details of the mashpoint framework. The implementation consists of three parts: (1) the applications themselves, which need to be data-centric in nature and built upon certain requirements, (2) a discovery service

that allows applications to look up other applications so that the user can pivot between them and (3) a means of communication between the applications and discovery service. In the following section, we discuss each part in detail.

5.3.1 Applications

In order to enable pivoting between applications, they need to be designed according to some specifications and rules. Our choice of specifications was motivated by a desire to make the integration of new and existing data-centric applications as painless as possible i.e. not to impose any unnecessary learning curves or restrict developers to use any particular technology other than what is currently in the Web technology stack. Thus, to link an application to mashpoint, the applications need to comply with the following principles:

- **Offer Data-centric features.** Each application in mashpoint must be such an application that offers interaction over data with identifiable entities. An application can hold multiple collections or grouping of identifiable entities - for example, be about People, Countries, Events etc. Applications, such as the ones in Figure 5.1 are typical examples of data powered applications that offer browsing over data about countries. In the data of these particular applications, each country is an identifiable entity.
- **Use of URIs as identifiers.** While the data underlying the application does not necessarily need to be in RDF, an URI needs to be present for each identifiable resource of data. For example, if the application uses data about countries, then each Country needs to be associated with an URI.
- **Be able to select multiple resources.** An application in this framework should typically enable selection of entities in order to be able to pivot with arbitrary selections of data. Selections of data can be provided in multiple ways. For example, items can be selected through filtering by providing various facets over data and/or allow arbitrary items to be selected. This selection of items will then be passed on as input to another application. In mashpoint, the current selection of entities in an application denotes the state of the application. In mashpoint we require each application to list the current entities in view through a `mashpoint` parameter in the URL. Figure 5.6 depicts the saved state in each application. Figure 5.6a, for example, depicts an application showing a single resource and a `mashpoint` that denotes this state. Similarly, Figure 5.6b shows the interface on a state with two resources. The state can also group entities (Figure 5.6c) in collections of entities e.g. an interface that displays data about both “Countries and “Currencies, show two groups of URIs. Whenever an application changes its

focus (for example a filtering operation occurs), the URL of the application which shows the state needs to change accordingly⁷.

- **Be able to represent multiple resources on input.** By having the state of the application explicitly stated in the URL, applications allow any arbitrary selection of URL identifiers to be used as input for that application. Thus, if an application displays data about country information, any subset of countries can be displayed on demand through dereferencing the applications URL with the mashpoint parameter listing the corresponding URIs.

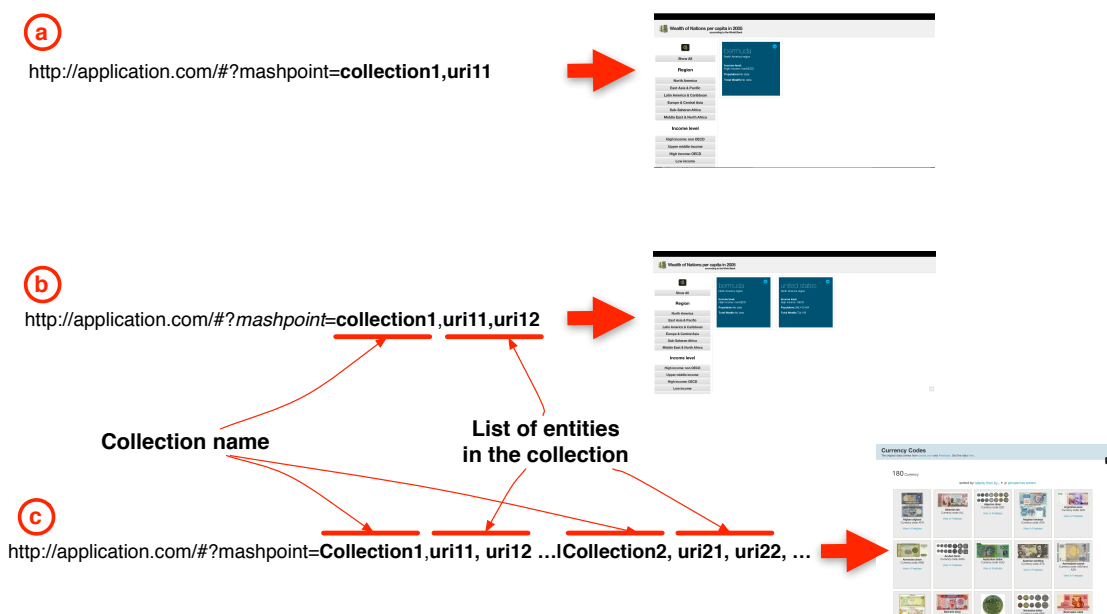


Figure 5.6: Preserving the state in a mashpoint-linked application.

The choice of URIs is also an important factor in the current implementation of the framework. In order to enable pivoting between applications we need identical identifiers across all mashpoint enabled applications. In our current instantiation of mashpoint we rely on Freebase⁸ as a service in which data used in applications need to be reconciled. Reconciliation is the process of defining that two things have the same meaning in a given context. In this context, data reconciliation means that entities described between two publishers need to be confirmed as meaning the same thing. For the mashpoint prototype we used Freebase⁹ as a reconciliation point for all the entities described in our applications. We would like to note that we have chosen Freebase for convenience reasons - Freebase and the support offered in Google Refine offer tools to quickly reconcile¹⁰ arbitrary data with Freebase concepts. While the data in the applications need to be

⁷While the approach of explicitly saving the state in the URL might not be scalable, approaches such as URL shorteners can mitigate any issues related to scalability due to URL length limits

⁸<http://www.freebase.com>

⁹<http://www.freebase.com>

¹⁰<http://code.google.com/p/google-refine/wiki/ReconciliationServiceApi>

reconciled with Freebase, it does not preclude using other data sources that already use established URIs. For example applications consuming Open Linked Data can use resources such as sameas.org¹¹ to either reconcile their data with Freebase or even use the service in real time (although the former is probably the preferred solution because of optimisation issues). In essence, it does not particularly matter which URIs we offer as reconciliation, since the framework requires just reconciliation of identifiers. Moreover, the architecture could also be redesigned in a different way - it could allow applications to use whatever URIs they see fit and try to reconcile them and do discovery in real time through the use services, such as sameAs.org. From a scalability perspective, however, a priori reconciliation provides a more optimised solution to the co-reference problem, then reconciliation on demand.

5.3.2 mashpoint Discovery Service

The mashpoint button provided in every mashpoint application opens the mashpoint window that shows applications from which users can choose to pivot to other applications. Finding applications which can be used to pivot from the current application is implemented through a discovery service for mashpoint-enabled applications. The discovery service is a repository that simply keeps a record about which URI identifiers can be represented in which applications. Applications, therefore, need to register themselves in the discovery service and “subscribe their URI identifiers. Registering with a set of URIs means that an application can represent and show data about any subset of identifiers it is subscribed to. Once registered, each application can communicate with the discovery service to find other applications that can take the current selection (represented through the URIs in its state) as an input. Figure 5.7 depicts this architecture. For clarity, the Figure shows URI identifiers represented with dots, squares and triangles to denote different collections of URIs found across different applications. For example, Application 1 is registered with the dot identifiers, which means it can take any subset of these identifiers as an input. Application 2 can either take any subset of dot identifiers or any subset of square identifiers as an input. Similarly, Application 3 can take subsets of square and triangle identifiers. These groups of URI identifiers are assigned by the application registering to the discovery service.

5.3.3 Pivoting Across Applications

In order to enable pivoting across applications, applications need to communicate and request information based on the current state of the application. Each application, therefore, communicates its state to the discovery service i.e. it sends the URIs that

¹¹<http://sameas.org/>

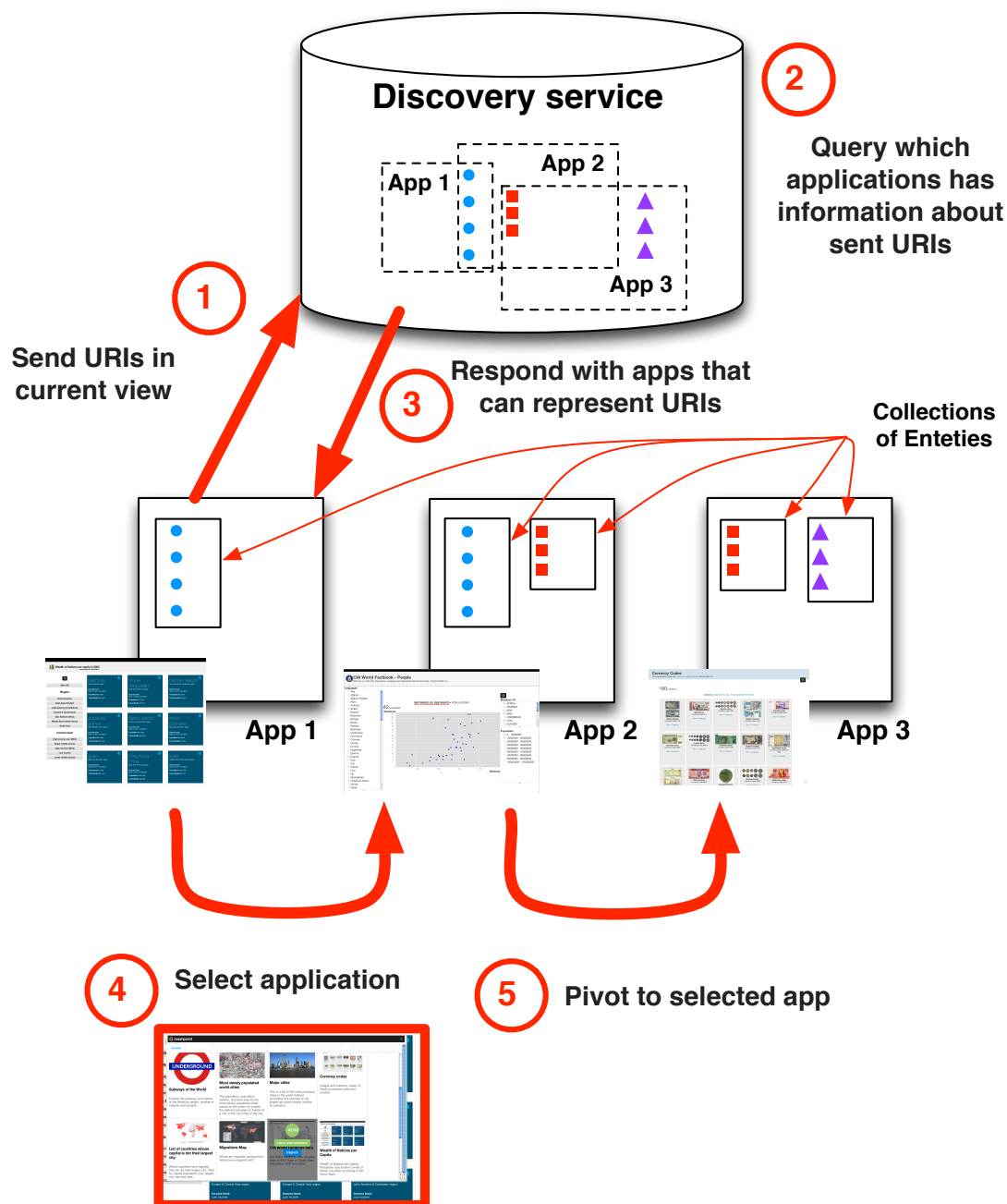


Figure 5.7: Architecture of the mashpoint framework.

currently represent the data viewed in the application, and retrieves back a list of applications that are able to receive those URIs as an input.

In order to facilitate this communication, each application in mashpoint incorporates a small JavaScript widget that is able to parse the URL for the URI identifiers and send them to the mashpoint discovery service (Figure 5.7-1) The discovery service then retrieves which applications can take the URIs as an input and sends them as a response with their states reflecting the identifiers in the request (Figure 5.7-2). The widget

in each application is a third party code that adds the mashpoint button, facilitates the communication with the discovery service and pops up the mashpoint window that suggests appropriate applications to users. We note that the discovery mechanism and widget may be omitted from an application. For example, there may be cases where a publisher of an application may want to offer pivoting to only a certain, predefined set of applications. Therefore, the publisher of the application can discover those applications once, and include them as regular links in the application. This removes the need for a third party discovery service; however it is now up to the publisher to keep the links to the other applications consistent with the current state of the application.

5.4 Evaluation

By pivoting with entities between data-driven web applications, the mashpoint framework aims at extending Web navigation to allow data-centric navigation when users need to perform data oriented queries. To evaluate the usability of formulating data-centric queries using multiple applications linked in the mashpoint framework we did two rounds of user studies. The first study was an exploratory think-aloud study designed to: (1) evaluate if mashpoint's data-oriented navigation was easily learnable, (2) assess if users found difficulties in comprehending different situations of data-centric navigation (i.e. situations when applications have different cardinality and between applications with different information concepts), and (3) surface any usability issues related to our initial prototype. Based on the insights gained in the first study, we created an improved version of mashpoint. We then launched a second user study to investigate the broader challenge of designing tools that will help users find the right applications when the set of available applications is large.

5.4.1 Data and Application Gathering

To do our studies we needed to have a reasonable number of applications linked to the framework. We did not need individual applications holding large amounts of data by themselves; rather only enough data so that filtering over that data in each individual application made sense. This also helped us mitigate any performance issues that can arise with scale. For the purposes of our study we relied on relatively simple sets of data centred on countries, major cities, currencies, and heads of state. The data around these concepts were also chosen to be data that most people are familiar with e.g. GDP, Population, Migration data, etc. We found a number of open source applications that we could easily modify and integrate in the framework. These included some of the example applications created with Exhibit (Huynh et al. (2007c)). We created plug-ins for client-side faceted browsers such as Exhibit and Isotope¹² to make these applications

¹²<http://isotope.metafizzy.co/>

mashpoint-enabled. We then used these tools to create additional applications from various sources on the Web, such as Wikipedia¹³ lists and its structured-counterpart DBPedia (Auer et al. (2007)). The applications were created to include different filters, different visualisations and have an overall different look and feel in order to give them resemblance as if coming from different publishers.

5.4.2 First User Study

5.4.2.1 Study Design and Procedure

For our first study, we recruited eight participants through an open email advertisement among graduate students and staff. Six of the students were male and two female. The ages of participants ranged 25-51. For their participation, each participant was given a 10 GBP gift card as gratuity.

We assembled a set of eleven different applications. Since the focus of our study was to grasp and understand the usability of various concepts of data-oriented browsing introduced in the mashpoint context, we needed to minimise the effect of any potential problems that might occur due to the discoverability aspects of finding appropriate apps to complete a certain task. Thus, we needed users who already familiar with all of the applications that were going to be used in the study. To accomplish this, participants were asked to go through an exercise, where they opened each application used in the study, and for each one they wrote down what sort of data was used in the application and, if provided, any options to do filtering and/or sorting. The participants were then allowed to use what they have written during the entire session. In order to familiarise users with the interface and concepts of browsing through apps, we ran a hands-on training session. During the training session participants were given five structured tasks with increasing difficulty. During the training session participants were allowed to engage in conversation with the examiner and ask any questions related to either the tasks or solutions. After completing the training session, participants were handed out an additional six tasks (five structured and one unstructured task). Both the training and examination set of tasks had the same structure (see Table 5.1). For example, the first two tasks needed to be completed with navigation only between applications that show data about a single collection of entities (Countries), and had the same cardinality (the applications had data about all countries). The third task required users to use applications that had different cardinalities (e.g. filtering by navigation to an application listing only EU countries), but conceptually about the same type of entities. The fourth and fifth tasks, required navigation between applications that used several different collections of entities. During the evaluation stage we recorded both the participants screen as well as an audio recording. After completion, the participants were asked to fill

¹³<http://wikipedia.org/>

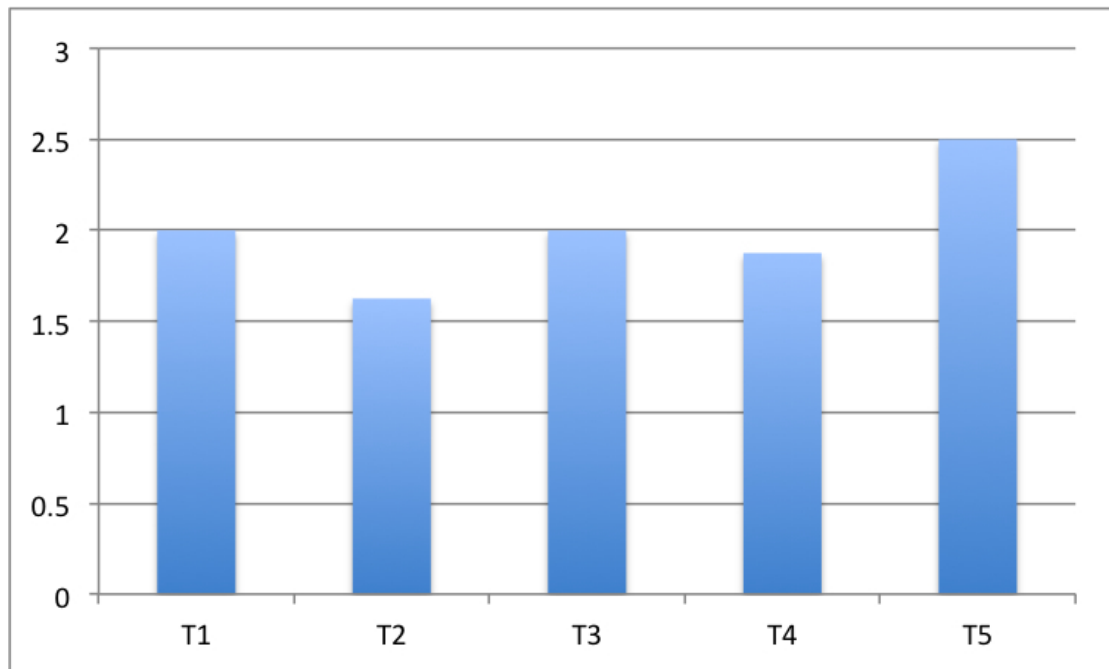


Figure 5.8: Ranking the difficulty of tasks in the exploratory user study.

in an exit survey and reflect on the tasks. The survey included demographic questions, as well as ranking the overall usability of interaction with multiple applications on a (1-7; 1 - Very Easy, 7 - Very Hard) Likert scale. We also asked participants to rate the perceived difficulty of each task (again on a scale from 1-7; 1 - Very Easy, 7 - Very Hard). Upon ranking the tasks, we asked participants to give an explanation about the given rankings. Additionally, we asked them for feedback on specific functionalities of the system.

5.4.2.2 Results

All participants were able to solve the tasks correctly. We gave the participants a time slot of 20 minutes to complete all five structured tasks (4 minutes per task); all participants completed their task before the allocated time was up (average time 15m 38s, standard deviation 3m 09s).

By far, the biggest bottleneck for completing tasks was finding the right application that had a particular filter, even despite the fact that users had gone through all the applications in the training session. This situation was apparent when either the filter or the application containing the filter was not an obvious choice. For example, participants took more time to solve the first task. This was because finding the filter for “African countries was difficult to locate, since the value was located in a filter named “Member of which included geographical regions as well as values such as membership in international organisations that made it difficult to spot. We also observed the number of times users back-tracked i.e. took a wrong step in solving the tasks. Overall, this happened only

Training Tasks		
No.	Task Type	Task
1	1-to-1 mappings with 2 apps	Find low-income countries and then display them geographically on a map.
2	1-to-1 mappings with more than 2 apps	Find high income (OECD members), then find out the ones that have a population between 10-20M and display migration patterns on a map for those countries.
3	Same concepts with subsets	Find which countries have between 5-10% CO2 emissions change between 2008-2009 and then find which of these are members of the EU.
4	Heterogeneous collections with 2 apps	Find countries with high corruption index (Index below 3 in 2011) and then find if any of these countries have a head of state that is among the top wealthiest heads of state.
5	Heterogeneous collections with more than 2 apps	Find dirtiest cities with an index of 40-50, then for those find which are low income countries and view those countries on a map.
Evaluation Tasks		
No.	Task Type	Task
1	1-to-1 mappings with 2 apps	Find African countries that have a population between 30-20M and then find their GPD per capita.
2	1-to-1 mappings with more than 2 apps	Find the worlds low-income countries, then find out the ones that have a population less than 10M and find the CO2 emissions data for those countries.
3	Same concepts with subsets	Find which countries have between 0-2% change in their corruption index between 2010-2011 and then find which of these are members of the EU and have an area size between 40000 ? 50000 km2.
4	Heterogeneous collections with 2 apps	Find countries with low corruption index (Index above 8 in 2011) and then find if any of these countries have a head of state that is among the top wealthiest heads of state.
5	Heterogeneous collections with more than 2 apps	Find the top most liveable cities that are ranked (0-10) and see if any of them are in countries where the capital is not the largest city and show these countries on a map.
6	Freelance task	Answer a question of your interest or explore some data using some of the applications in mashpoint.

Table 5.1: Training and evaluation tasks used in the user study. Column 2 shows the type of operations involved in completing each task.

on three occasions in the entire study session. Back-tracking occurred either when a participant did not have a clue about which application to choose to do a particular filtering, or if the participant chose a wrong filtering option which in turn caused the task to be incomplete. In one particular instance, a participant (P7) made an error in Task 3 when trying to find the EU countries from a set of countries with a 0-2% change in the Corruption Perception Index. The participant had selected the wrong filter in the first attempt, which yielded a set with countries, none of which were members of the EU. Since the set of countries did not include any EU countries, when the mashpoint window dialog was opened, the application with data exclusively about EU countries did not appear in the mashpoint dialog window. The participant paused and then tried to re-examine the task and realised the error. The participant pointed out that showing the EU application in the previous attempt with zero possible entities would have made him realise the error much quicker. Surfacing this kind of information can also be used to indicate to users that a different approach to filtering might have yielded additional available combinations with other applications. During the open task, users wanted to do comparison between chains of applications, however with the current prototype they had to reopen all the applications in the chain if a different filter was selected in an application that was in the beginning of the chain. Thus, some users commented on the need for a better support in iterative query refinement.

In the exit survey each participant was asked to reflect on the study and tasks that were handed out. On a Likert scale of 1-7 (1 - Very Easy, 7 - Very Hard) judging the difficulty of mashing data using multiple applications in the mashpoint framework, participants gave an average grade of 1.5 (half of participants gave a mark of 1 and the other half gave a mark of 2). We also found that there was no substantial difference in the perception of difficulty in respect to different tasks (Table 5.1). Task Five was given the highest difficulty rating (Average 2.5); however only one participant specifically mentioned it specifically in the exit survey stating:

“Initially I was confused about the city/country tabs [in the mashpoint window], but rereading the task, it should have been clear.

Other perceptions were described by participants when explaining the ratings they have given for a task. For example one participant noted:

“Initially, Task 3 seemed more challenging, because it seemed more numerical. In actual fact, the presentation of the granularity of filterability of datasets made the task much simpler than I originally supposed.

When asked to comment on potential improvements to the system, three of the participants wanted to have some way of searching for applications when the dialog opens

with two of them pointing out that a list of possible filters might be helpful. Three participants wanted support for iterative query refinement or the ability of changes in filters to propagate through a chain of applications. During the exploratory task, users also wanted see how individual entities related across applications. When we suggested that a user can filter for a particular item and re-render the trail again, the participant responded back by stating that some sort of highlighting an item and propagating the result through highlighting corresponding items in the trail would make comparison tasks much easier. One participant noted that some data pertained to different time periods and suggested that some restrictions or notification should be put in place to highlight such potential inconsistencies.

5.4.3 Second User Study

Based on the feedback we got from our exploratory study we improved mashpoint in two ways¹⁴: (1) we added support for iterative query refinement and (2) we added application search capabilities. In order to add support for iterative queries we created a web application that wraps applications in mashpoint in a browser-like experience and allows a trail of pivoting operations to be refreshed every time a filtering operation occurs in some application in the trail. Thus, if we have a situation where a user opens four applications through pivoting, and wants to filter the results in the first application, then all the other applications will reflect that filtering operation. In the initial screen of mashpoint browsers, a user can search for mashpoint-enabled web pages. Upon selecting an application, the application opens in a new tab similar to how we open new pages in a browser. With every data-oriented operation the navigating application is displayed as a separate tab. If users go back to an application in the trail and select a different set of items, the entire trail of applications is refreshed and the filtering results are propagated through the trail. We currently support only one trail at a time; if a user does a new data-oriented operation from an application in the middle of the trail the entire trail from that point onward is deleted and a new sub-trail starts. In the future, a browser design can be constructed to support multiple trails. This would go a step further in better supporting quick iterations of different queries.

In respect to search options, we wanted to test if only keyword search is sufficient when trying to find an application or if filtering over metadata about the applications would be a preferred method for discovering applications.

5.4.3.1 Study Design and Procedure

For our second study, we designed two variations of the browsing interface. The first one (Browser Version) included a keyword search option, which queried over an index of

¹⁴see <http://mashpoint.net/browser/> and <http://mashpoint.net/navigator/>

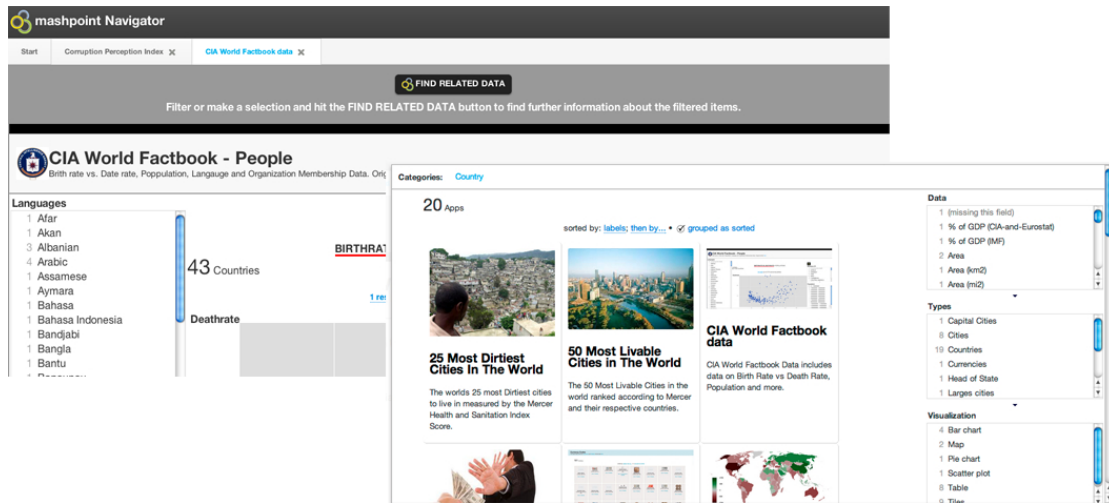


Figure 5.9: Mashpoint applications embedded in a browser-like application and extended with faceted search.

the application descriptions, and text values in data. In the second version (Navigator Version), we used the metadata gathered from the training set in our first study to create a faceted interface to query for application. The facets included the type of data the browser already dealt with (e.g. Countries), a facet about the facets of data in each application (e.g. Population) and a facet which shows how data is displayed (e.g. Table, Bar Chart etc). Additionally we increased the size of the application set to 20; the rest of the metadata for new applications were provided by asking three other people to go through the same exercise of describing the applications as the participants in the first study. In cases where a facet was too vague (e.g. 2008/2009 change), we added the data type as well. For the study we recruited 22 undergraduate students (students of Economics and Social Studies), 10 female and 12 male. Their ages ranged from 19 to 25 (average age 21). We choose undergraduate students because of their frequent use of Web technologies, and in particular we chose students in economics because we wanted to have participants that would be interested in the data that we were using in our applications (economic and social indicators about countries). Working with participants who have an interest in the data has been found to be especially important in search usability studies (Borlund and Ingwersen (1997)). Because we wanted to mitigate any carry-over effects of learning the applications when using the tool, we opted for a between-subjects study. Thus, participants were divided into two groups of eleven members. Each group had five females and six males.

Similar to the first study, the participants went through a training session, in which participants were familiarised with the concepts of mashpoint and the browser-like interface. Unlike the first study, we did not allow users to view or browse through the application set; during the training session we used applications that were not used later in the evaluation tasks. We then gave the users four structured tasks to complete with

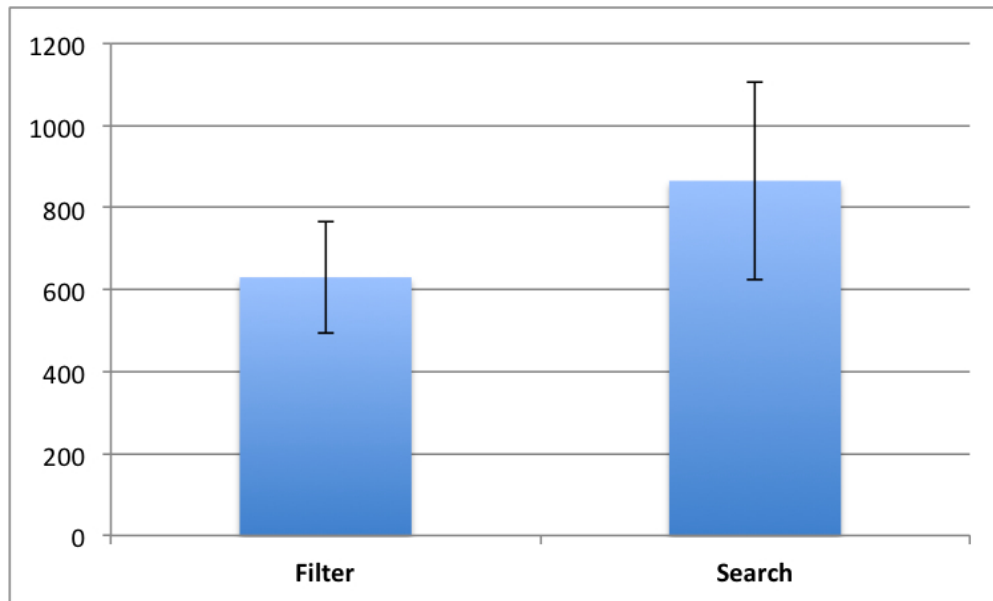


Figure 5.10: Average time it took to complete the tasks with Navigator Interface versus the Browsing interface.

no time limits. The tasks were the same in concept and structure to the first user study. During the evaluation session we kept a video record of the screen. After finishing, each participant was asked to fill in an exit survey.

5.4.3.2 Results

We ranked solutions of tasks into three categories: (1) solution is correct, (2) solution is incorrect, and (3) solution is partially correct (e.g. when a user chose the wrong filter). Overall 62 (70.45%) tasks were answered correctly, 23 (26.15%) were answered partially and 3 (3.40%) were incomplete. There was no difference between the groups in terms of completing tasks correctly. The group using the Navigator version, however, completed the tasks in an average time of 630 seconds, with a standard deviation of 135 seconds. The group using the Browser version completed the tasks in an average time of 835 seconds with a standard deviation of 241 seconds. These results are statistically significant ($p < .05$).

Again, in the exit survey, each participant was asked to reflect on the study and tasks. We first asked participants of both groups to judge the overall difficulty of mashing data using multiple applications in the mashpoint framework. To carry out a more sensitive reading of the result we switched to a Likert scale of 1-9 (1 - Very Easy, 9 - Very Hard). The results are shown in Table 5.11. Because the difficulty of tasks had increased, and since participants were additionally burdened with the task of finding the appropriate applications, we got worse scores on the overall difficulty when compared to the feedback we got from users in the first study. However, the rating by the participants using the

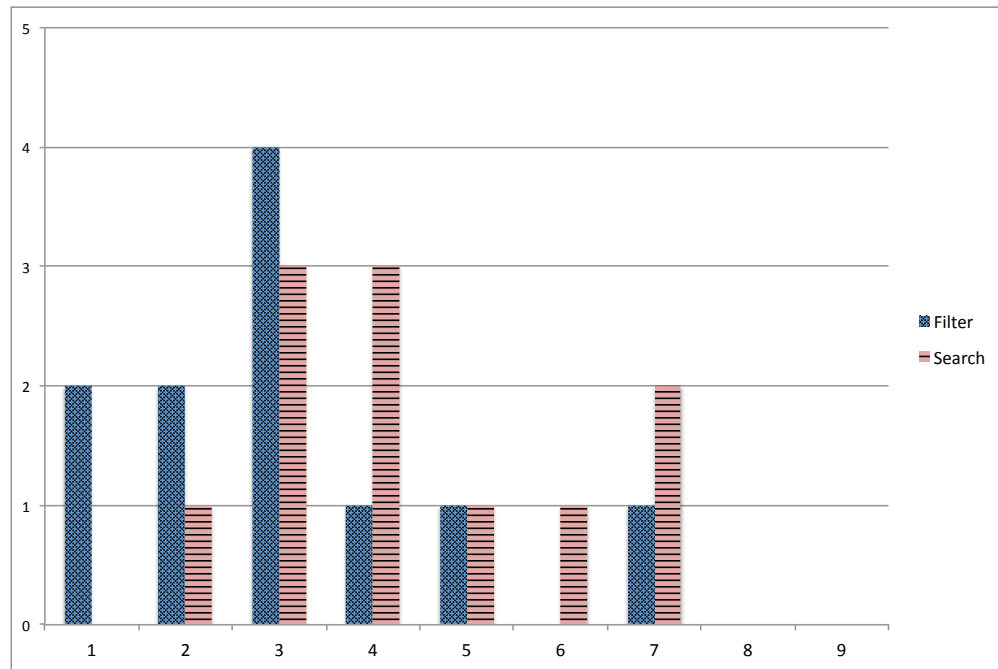


Figure 5.11: Evaluating the difficulty of mashing data using multiple applications in the mashpoint framework between the two groups using a Likert scale. The Figure shows how users from both groups ("Filter" and "Search") rated the difficulty of accomplishing their task on a scale of 1-9 (1 - Very Easy, 9 - Very Hard).

Navigator version was more favourable than by those using the Browser version, with the former giving an average score of 3.09 and the latter giving an average score of 4.36. From observing the sessions we noticed that the faceted browser was preferred over keyword search.

5.5 Enriching Unstructured Content in Websites

So far the applications in mashpoint exhibited two characteristics: (1) they were all data-oriented applications, and (2) they needed to comply with several principles in order to exchange entities and allow pivoting to other data-oriented applications. Some applications on the Web, however, are not data-driven and web pages have only recently begun including information about entities found in the content of a web page. To showcase the benefits of having structured information available on demand from existing web pages on the Web we deployed a tool that allows mashpoint applications to be surfaced based on entities extracted from unstructured pages. Figure 5.12 illustrates an example of structured data surfaced from an article found on the CNN¹⁵ website. To use mashpoint applications in unstructured webpages, a bookmarklet is added as a bookmark on a users browser. When the user presses the bookmark, the bookmarklet application

¹⁵<http://www.cnn.com/>

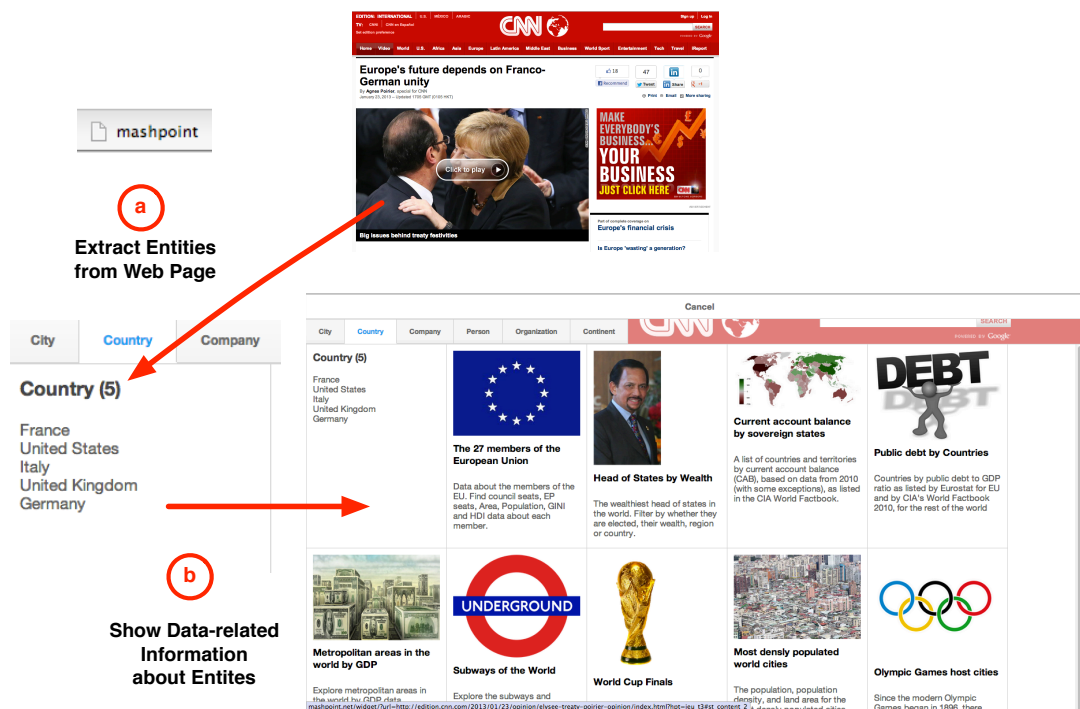


Figure 5.12: Using mashpoint to show structured information about entities in unstructured pages.

tries to extract entities from the text in the currently open website. The extraction is done using name-entity recognition services such as OpenCalais¹⁶ and AlchemyAPI¹⁷. Once the entities are extracted, a list of applications is retrieved based on those entities. Thus, if we suppose the user reading a web page article about Germany and France (as shown in the CNN article in Figure 5.12) has a question whose answer requires structure data, this information can easily be accessed without the need to use search engines in order to find the information on other, unstructured pages.

5.6 Summary

In this Chapter, we presented a framework for linking applications on data level, in effect enabling browsing of Linked Data through the lens of multiple distributed applications. On a conceptual level, the approach solves some problems that generic browsers currently experience, by extending data-mapping approaches to data-sharing capabilities. Our studies show that execution of complex data-centric tasks are easily completed using mashpoint, and that finding suitable applications to solve a data-centric task can be supported.

¹⁶<http://www.opencalais.com/>

¹⁷<http://www.alchemyapi.com/>

Chapter 6

Discussion: Implications, Challenges and Future Directions

As one can observe from the tools presented throughout this thesis, designing tools over Linked Data often requires balancing and making trade-offs with the ability to deploy the tool over Linked Data with ease, the ability of the tool to access arbitrary data, and the usability of the tool. For example, currently tools that allow arbitrary access and ability to navigate the entire Linked Data space are only those that are able to retrieve data through dereferencing. Since federated queries are still relatively slow in order to be at the needed level of responsiveness in an end-user tool, these tools cannot provide advanced filtering and sorting functions usually required of database technologies in the background. Thus usability is sacrificed at the expense of universal access. If a browser restricts to only navigating data in datasets that expose a SPARQL endpoint, it can provide advanced features that require database technologies as well as universal access to any SPARQL endpoint, but it would not be able to easily query, navigate, or combine data found in multiple datasets. On the other hand, as our mashpoint framework shows, large number of data-centric queries can be answered using linked applications with the usability being generally on par with exploring information in custom made applications; however the application abstraction results in a reduced flexibility in data manipulations - i.e. the user is only offered ways to filter or interact with the data offered by individual applications.

Figure 6.1 shows the landscape of tools that enable interaction over Linked Data. The X axis indicates the effort needed to deploy the tool over Linked Data. For example, at the start of the spectrum in the X axis are custom-made applications, since each application requires the effort of making an application from scratch. Configurable browsers require less effort because they can be adapted and configured over data sources with no or minimal amount of programming. Generic data browsers assume no need for customization of any kind. The Y axis indicates ease of use for the tool deployed by

the end user. The size of individual dots denotes the relative ease with which tools can access data through navigation. As the figure illustrates, tools such as Tabulator, can access Linked Data on demand with no required configuration and can add or navigate to any other dataset available as Linked Data. Tools, such as Visor and Parallax, on the other hand, can access any dataset and provide advanced query features. Data-mapping approaches and custom made applications need to be built or deployed over predefined datasets, and generally do not allow navigation or adding any new data on demand. With mashpoint we began to address the limitations imposed by data-mapping approaches and custom-made applications.

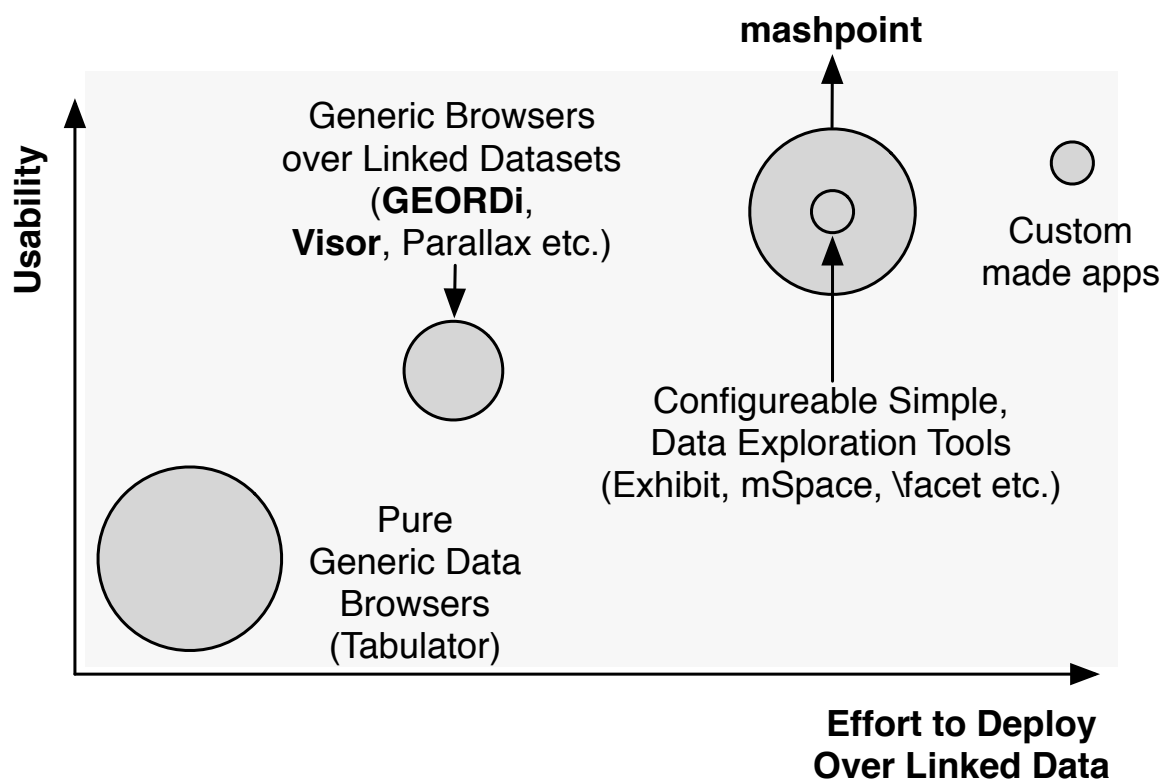


Figure 6.1: Landscape of data-centric tools over Linked Data.

Linked Data is still an emerging method of publishing data on the Web, one that will still evolve as technical challenges are overcome, as well as through social processes that will inevitably cherry pick, adapt and revise certain pieces of the technology when adoption becomes more mainstream. In this Chapter, we review some of the implications of the tools and approaches proposed in this thesis, and discuss areas where future directions might influence the design of tools providing data-centric interactions over Linked Data.

6.1 Generic Data Browsers

In this section we discuss two issues that influence the design and properties of generic browsers. First we explore the relation between usability in generic browsers and the use of common standards or practices in publishing Linked Data. Second, we discuss the issue of exposing data heterogeneity to end users consuming Linked Data.

6.1.1 Establishing Standard Ontologies or Vocabularies

As we've seen throughout this thesis, current generic data browsers must assume very little about the underlying data. These minimal assumptions are partially the reason behind poor usability in generic data browsers. While generic browsers cannot provide solutions that domain-specific applications can, adopting very simple minimal conventions or the use of minimal ontologies when publishing Linked Data can have a big impact on usability in generic data browses and would contribute to improving Linked Data quality in general. For example, one can imagine having quality validation of Linked Data if it satisfies predetermined basic requirements: for example that all entities have and use standard properties for labels, descriptions, images etc. Standard schemas can also be used in more specific areas, for example for publishing data about statistics etc. We have seen some work around establishing commonly used vocabularies, for example, VOID for standardising the publishing of information about datasets, SCOVO, an ontology for publishing statistical information, and SKOS¹ for organising hierarchies, thesauri, and classification schemes (Alexander et al. (2009); Hausenblas et al. (2009)). However, most of these efforts still have very limited success. The key to having such specifications lies in having applications which are ready to pick up published data. Recent initiatives, such as schema.org² and Facebook's Open Graph take this approach - each standard has been introduced after a clear application of data has been suggested as an incentive for the publisher to use the format to embed metadata in their webpages.

6.1.2 End Users and Data Heterogeneity

At the beginning of this thesis we stated that the purpose of this thesis was to investigate end-user interaction over Linked Data for tasks that required data integration. We also limited the scope of this thesis and were left dealing with heterogeneity, structural heterogeneity in particular. As one can also notice from our analysis in Chapter 3, no generic data browsers address the problems of structural heterogeneity directly. While there are many cases where using Linked Data would be beneficial and can be consumed

¹<http://www.w3.org/2004/02/skos/>

²<http://schema.org/>

without any need to solve problems of heterogeneity, there might be cases where structural heterogeneity needs to be resolved in order to use the data. Thus, when designing data browsers or any other type of data-centric tools that use Linked Data to aggregate data from multiple sources, there is always a debate whether data heterogeneity should be exposed in the interface and whether supporting the users with tools will allow solving heterogeneity problems on-demand, or if heterogeneity should be resolved in the background and never be surfaced to users. This thesis argues that, although tools for solving heterogeneity should be available for expert users, most end-users should not be exposed to heterogeneity issues. Solving heterogeneity is often a complex task that requires a workload equivalent to or in some cases even greater than the task for which the data will eventually be used. Even in interfaces designed with casual end-users in mind, such as Potluck, they still have very limited capabilities. More complex tools, such as OpenRefine (previously known as Google Refine), still require sound knowledge of at least writing small code such as spreadsheets macros (see Figure 6.2). Such knowledge cannot be assumed for large portions of end users, and especially casual end-users with no technical or data-related knowledge. Advances in more automatic approaches in solving heterogeneity, might facilitate bringing humans in the loop, however, as this thesis shows, there is a lot of low hanging fruit in using Linked Data to aggregate sources that do not need to be reconciled for structural heterogeneity.

6.2 Exploring Unfamiliar Datasets

In Chapter 3, we observed that one of the biggest problems of consuming Linked Data is interrogation and exploration of unfamiliar datasets. With Visor, we have shown one method of improving exploration of unfamiliar datasets. However, many functionalities and interaction techniques may exist in improving exploration and finding data in unfamiliar datasets. Researching some of these techniques, however, also depends on the capabilities in the backend. Currently, the query language for RDF, SPARQL is very limited for providing basic query functionalities. Upcoming versions of SPARQL, such as SPARQL 1.1³ offer capabilities such as finding property paths⁴, which allows finding potential paths in data directly through a SPARQL query. This allows exploratory queries that recommend paths in a graph, such as the ones introduced in Visor, but at a much higher scale. How these would be represented or utilised by end users is still an open question. In Visor, we approached this problem by using an over-first details-on-demand visualisation approach. Improved visualisation tools might provide even better instances of the overview-first details-on-demand approach. On the other hand, other approaches might also be viable. For example, recent research into verbalisation of SPARQL queries might provide a way for a natural language approach to representing property paths (Ell

³<http://www.w3.org/TR/sparql11-query/>

⁴<http://www.w3.org/TR/2010/WD-sparql11-property-paths-20100126/>

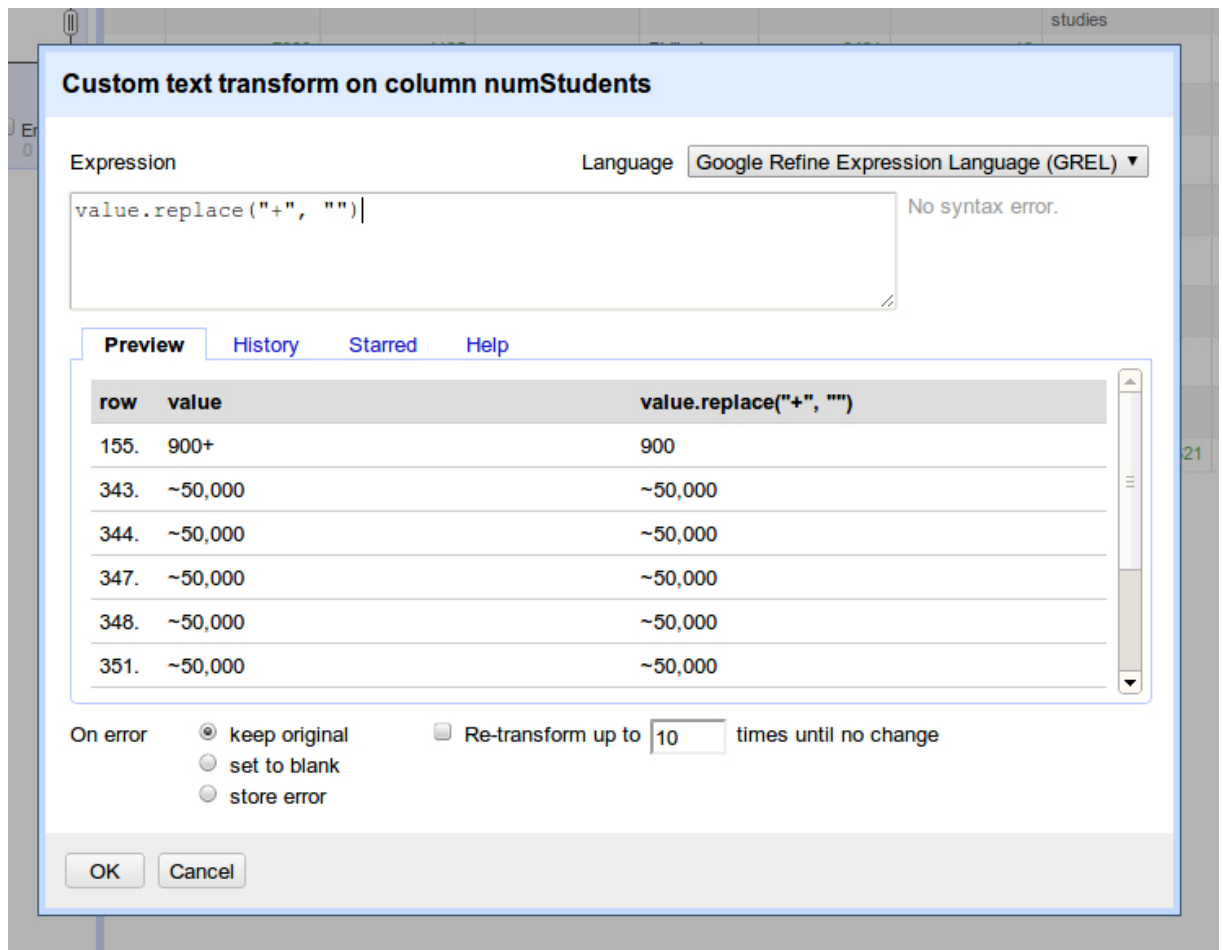


Figure 6.2: Data transformation from two sources in OpenRefine using macros. The Figure shows transforming numeric data by aligning different representations of rounding.

et al. (2012)). Recent development efforts to explore Facebook's social graph also use natural language to support structured queries (Figure 6.4). Google is also moving in similar areas with its Knowledge Graph (Figure 6.5). For more data-related tasks, tools similar to Google's Google Square (Figure 6.3) prototype might use these approaches to produce a more accurate way of producing spreadsheets, which currently draws its data based on search heuristics rather than structured queries.

6.3 mashpoint

In this section we examine the problems of co-reference in using mashpoint, and examine the cost/benefit structures of mashpoint versus Linked Data technologies in general.









Item Name	Image	Description	Weight	Height	Group
<input checked="" type="checkbox"/> Chihuahua		Do not let the Chihuahua get away with things you would not allow a large dog to do (Small Dog Syndrome), such as jumping up on humans. ...	6 lb	6-9 inches	Toy
<input checked="" type="checkbox"/> Maltese		Do not allow these dogs to develop Small Dog Syndrome , human induced behaviors, ... Today, the glamorous Maltese is an adored pet and sought-after show dog	8 lbs	18 to 24"	Toy
<input checked="" type="checkbox"/> Bichon Frise		A Bichon Frisé (French, literally meaning curly lap dog) is a small breed of dog of the Bichon type. They are popular pets, similar in appearance to, ...	7-12 lbs	9 - 12 inches	Non Sporting
<input checked="" type="checkbox"/> Affenpinscher		Description, The Affenpinscher is a small dog with a harsh, shaggy coat, and longer hair all over the face. It is a smaller version of a working terrier and ...	7-10 lbs.	10 - 15 inches	Toys
<input checked="" type="checkbox"/> Brussels Griffon		Brussels Griffon Breed Standard. Toy Group. General Appearance A toy dog, intelligent, alert, sturdy, with a thickset, short body, a smart carriage and ...	6-12 pounds	7-8 inches	Toy
<input checked="" type="checkbox"/> Pomeranian		You may find Pomeranian puppies for sale and Pomeranian dogs for sale from quality dog She Is Soo Small . She Has A Gorgeous Little Baby Doll Face, ...	4lbs	7-12 inches	Toy
<input checked="" type="checkbox"/> Havanese		The Havanese gives a rugged impression of a little dog , it is sturdy, and while a small breed, it is neither fragile nor overdone. ...	7-13	8-11	Toy
<input checked="" type="checkbox"/> Australian Terrier		"Dedicated to the Advancement of Quality, Purebred Australian Terriers ". Founded	12 - 14 lb	10 in	Terrier

Figure 6.3: The Google Squared tool. After a search results in a list of entities, users can add columns by typing in column names, which are used as a keyword to search for those properties in the initial list of entities.

6.3.1 The Co-reference Problem and Approximate Semantics

The early efforts of the Semantic Web were, to a large degree influenced by the vision of automating many processes done by machines, replacing the human who usually added the required intelligence in the process loop. Arising from the field of formalising knowledge and knowledge representation, early Semantic Web research often aimed at describing data in machine readable formats that left no space for ambiguity. This led to some initial criticisms of the Semantic Web, stating that it aimed to describe the world in a unified way, and that such a proposition is impossible because many descriptions were influenced by the context in which they were used. For example, should a common identifier about the state of Germany be used to describe present Germany and Germany of World War Two? What about describing Germany, when the country was divided into two separate states? When are two things semantically equivalent? In some situations, the equivalence might hold true; however there might be cases when the two should be distinguished. Some Semantic Web researchers have argued that equivalence links (i.e. `owl:sameAs` links) represent a strict semantic equivalence,

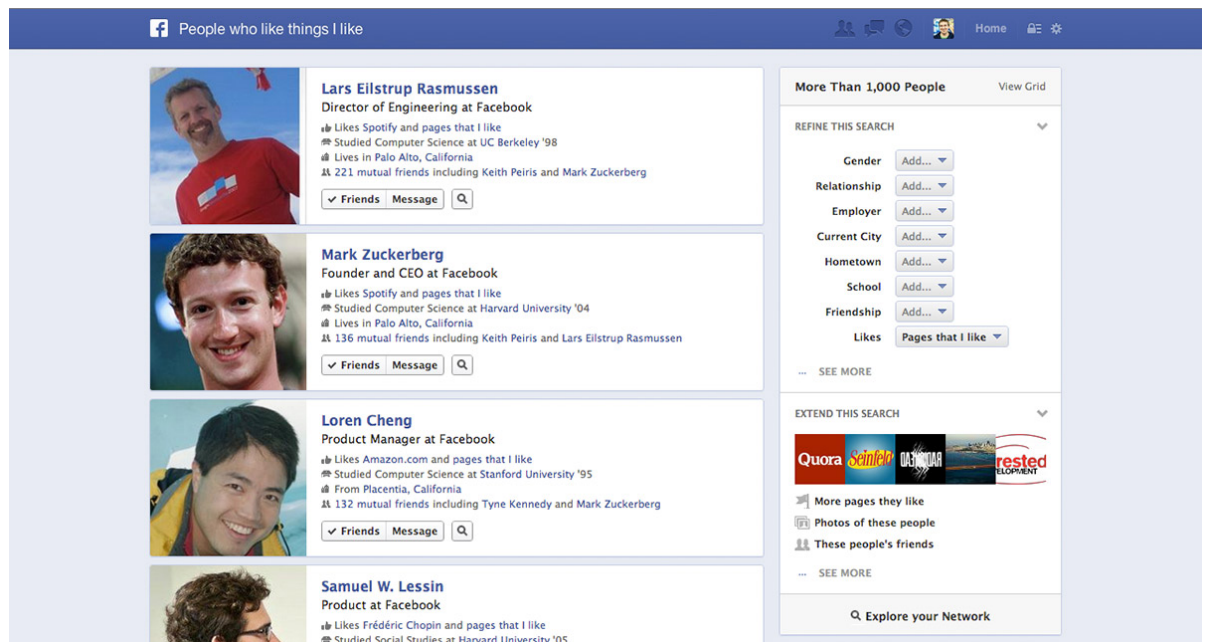


Figure 6.4: Facebook's Graph Search.

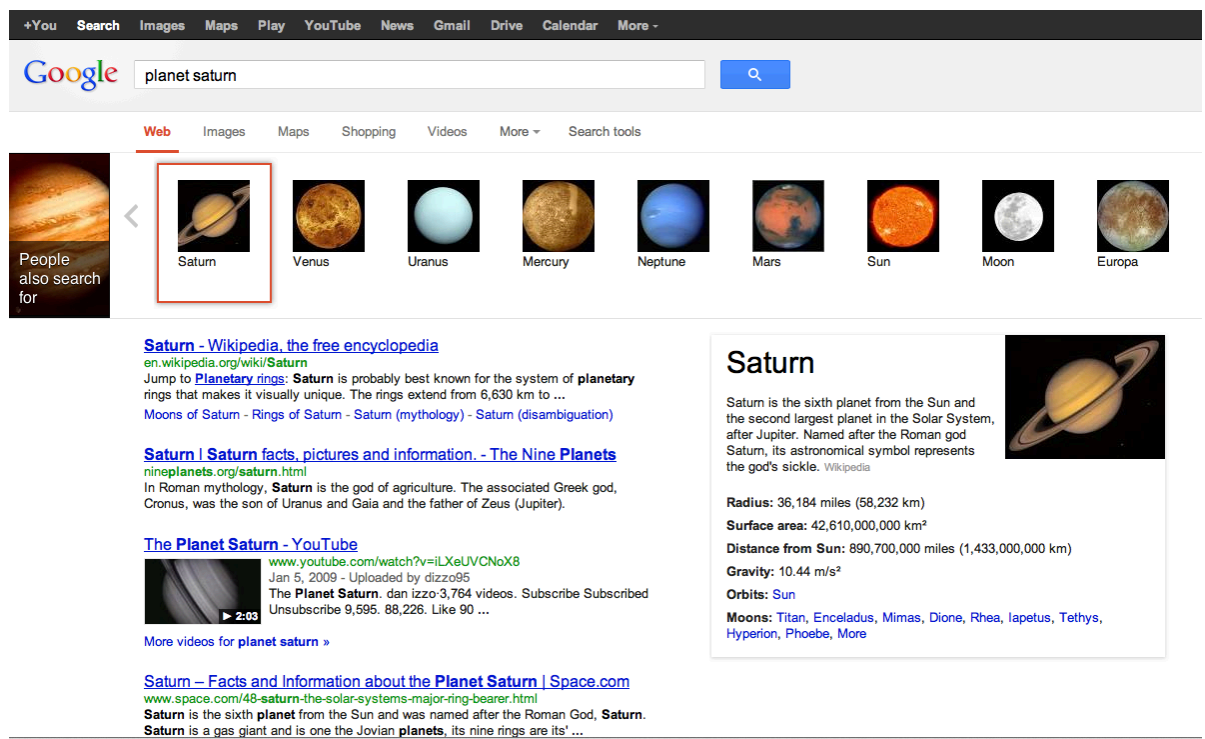


Figure 6.5: Google's Knowledge Graph showing structured information about planets.

while others have pointed out that there might be multiple representations and meanings for the particular property (Halpin et al. (2011)). Tools such as sameAs.org⁵, which is

⁵<http://sameas.org>

a website that mines and stores equivalence links on the Web, for example, holds not a single one, but multiple repositories of equivalence links. Thus, each individual data publisher states if the entities described in their datasets are equivalent to other entities in other datasets. This thesis argues that if a loose definition is used, then users can decide the correctness of the equivalence statement depending on the context. This was one of the issues we were confronted with in mashpoint. If data-centric browsing on the Web will become a mainstream technology, problems of context might also surface in tools such as mashpoint, since the bases of mashpoint is that applications use canonical identifiers in order to communicate. However, we speculate that it would be better for users to be able to discover potentially relevant data sources and then decide if the data obtained through pivoting is right for the given context of exploration.

6.3.2 Cost/Benefit of Using Linked Data Technologies

6.3.2.1 Low Barrier for Entry

One of the biggest barriers of adopting Semantic Web technologies is its high cost for adopting these technologies. Following the description of the mashpoint framework, we can notice that none of the applications is required to directly operate or deploy its data as live Linked Data, nor does the framework require an application to use RDF data directly. We believe this fact to be an added strength to the framework, since it only lowers the barrier for linking new applications by not mandating or imposing any particular data model on the user. This is not to say that the applications using this framework cannot use standard data-models such as RDF. In fact, as we pointed out in Chapter 5, one way of defining mashpoint is to view the applications as high-level lenses over graphs of data. The applications encapsulate views over data, and the relationships between the data are hidden within the individual applications. Thus, applications can choose to use either RDF or any other data model and pivoting takes place where these lenses overlap. Thus, mashpoint can also be one case in point for having Linked Data without the need of RDF. The RDF data model might be more expressive and in time may become an adopted standard for publishing data on the Web. At this time, however, we argue that economic factors such as lowering the costs and showing immediate benefit should be the challenges that we tackle in order to incentivise users to adopt semantic technologies, even if in the beginning these are only confined to the simplest cases, such as providing universal identifiers and reconciling against canonical URIs.

6.3.2.2 Incentives for Publishing and Linking Applications

During the last 3 years, the Linked Data community has been advocating data publishing using Linked Data standards, and has promoted the use of these standards as a quality

indicator for data available on the Web⁶. However, the benefits of publishing Linked Data and linking to other remote data sources remain elusive for most data publishers and consumers outside the community. Often the results are data repositories that are rarely used and provide sparse linkages to other remote datasets. This lack of immediate value for the effort of converting data as Linked Data can thwart many potential adopters of these technologies. With mashpoint we attempt to target this problem, particularly in providing incentives for linking data. From a socio-economic perspective we believe that publishers of data-centric applications would only increase the value of their applications by allowing users to find useful, related data without changing the original application, similar to how they link to other Web sites. By requiring publishers to reconcile their data we already promote the use of URIs in their datasets, thus promoting semantic technologies, while showing the immediate benefit of being able to pivot and suggest related data to the application users.

⁶<http://inkdroid.org/journal/2010/06/04/the-5-stars-of-open-linked-data/>

Chapter 7

Conclusions and Future Work

The main motivation behind this thesis is that Linked Data, as a medium for published data on the Web, should be accessible to non-technical users, which can use the properties of Linked Data, to support data-centric tasks that require combining structured information from multiple sources. In this Chapter, we summarise the contributions of this thesis and continue with possible future directions from this line of investigation.

7.1 Summary of Contributions

In this work we have presented the following contributions:

- **Design Process for Data-centric Interfaces over Linked Data.** In Chapter 3 we presented a design process for data-centric interfaces over Linked Data. The design process started by suggesting use-cases, stockholders, going through a requirement elicitation exercise, an analysis of existing generic data browsers, and finally a prototyping effort to elicit a list of challenges in generic data-browsers. Creating a design process is a first attempt in the field to systematically analyse the requirements and challenges of data-browsing interfaces over Linked Data. It provides future designers with a map of needed areas and challenges for designing future data-centric interfaces, as well as basis for comparisons.
- **Multi-pivot approach.** In Chapter 4, we presented the multi-pivot approach - a novel approach that aims to improve and mitigate some identified challenges in using purely navigational models for data exploration in graph datasets. The results of our study suggest that users can explore large unfamiliar graph datasets, and that the additional flexibility of multi-pivot approach is preferred over purely navigational approaches. Additionally the overview-first details-on-demand approach, suggests that ontology level browsing and instances on demand is a viable alternative to exposing only instance data to users.

- **A framework for extending pivoting to applications.** In Chapter 5, we introduced mashpoint, a framework aimed at extending pivoting with data over data-powered applications. The approach is the first attempt to combine the inherent usability of data-mapping approaches, with navigational extensions which allow data to be passed from one application to another. The user studies show that data-centric interactions can be carried out without the need of generic data tools. Additionally, our second user study suggests that a data space populated by multiple mashpoint-linked applications is searchable i.e. users equipped with interfaces and proper search tools allow the user to find needed applications to complete a data-centric task.
- **Lowering the cost of using Semantic Web technologies.** Finally, our mashpoint approach shows that even using the basic technologies in the Semantic Web technology stack only (such as URIs) one can provide affordances that were not previously possible on the Web. By using only URIs, we have shown that many of the benefits with data navigation and aggregation that are needed for data-centric interaction can be achieved without requiring from the publishers of data-centric applications to include more complex technologies in the upper level of the stack. This suggests that some of the benefits of semantic technologies come at a much lower cost.

7.2 Future Work

The Web has transformed the world in which we live and work. Work presented in this thesis presents a way of transforming the Web by extending it to support data-centric features, thus allowing access to end users with various needs and skill sets to utilise structured information published on the Web. In this section, we describe new areas of research and propose some potential projects that can build upon the work presented in this thesis.

7.2.1 Collaboration

This thesis presented several examples of data-centric interactions on the Web, and solutions for supporting such data-centric interactions over Linked Data. As one can notice, however, data-centric interactions can often be a task; tasks that require a substantial amount of effort and time to explore, gather and analyse information. These sorts of tasks differ from short-term tasks such as fact-finding tasks or transaction tasks. A potential way to improve efficiency can be to capture and utilise knowledge and processes in previous user sessions and expose them as solved tasks to other users attempting similar tasks. For example, whenever a user explores a dataset and generates a spreadsheet,

the spreadsheet of data can be potentially saved, so other users, looking for similar data, can use the data as it is or use it as a starting point and further modify it. This would prevent other users who have similar tasks starting from scratch. One can envision capturing other information, such as visualisations and data transformations. In effect, a collaborative effort where users can build upon each others solutions, saving and exposing their results of data-centric interactions can be seen as a community-based effort of generating lenses or other predefined views of data.

7.2.2 Crowdsourcing Data-oriented Tasks

Human Computation, crowdsourcing and crowd-powered interfaces are parts of an area that investigates how to bring people as active participants in the computation loop, in order to solve AI hard problems that are currently unsolvable by computers (Quinn and Bederson (2011); Kittur et al. (2008); Bernstein et al. (2010)). These relatively novel areas have only begun with applications in areas such as the Semantic Web/Linked Data. Several examples include tools for solving tasks, such as ontology alignment and co-reference problems (Sarasua et al. (2012); Yang et al. (2011)). One can investigate a number of problems identified in this thesis, to ascertain if crowdsourcing is a viable solution, and if so what are the challenges that need to be addressed in order to come up with a solution. Potential areas include improving dataset information by using crowd-based solutions, or crowdsourcing data-centric queries to generate data spreadsheets. For tools such as mashpoint, existing approaches to crowdsource reconciliation can begin addressing the challenge of reconciling large amounts of data. Other areas might include grouping of applications that provide related content for a set of entities.

7.2.3 Supporting Domain-specific Areas

The adoption of Linked Data technology ultimately depends on having applications and true use cases that can only be supported by using Linked Data technologies. There are many domain or task specific areas where data-centric interactions can be applied on a less generalised scale. For example, content curation on the Web is gaining ever bigger traction, showcased by tools such as Pinterest¹ and Clipboard². These existing systems assemble pieces of information on the Web; however exhibit very little metadata. One might investigate how enriching metadata webpages can support more effective content curation. Other domain-specific areas, such as bio-medical informatics, also extensively use complex data domains to collaborate and gather information. Specialised data-centric tools supporting these areas might provide better solutions than a general data access tool.

¹<http://www.pinterest.com>

²<http://www.clipboard.com>

References

- Ben Adida and Mark Birbeck. **Rdfa primer**. Technical report, W3C, October 2008.
- Kieth Alexander, Richard Cyganiak, Michael Hausenblas, and Jun Zhao. Describing Linked Datasets - On the Design and Usage of void, the 'Vocabulary of Interlinked Datasets'. In *Proceedings of the Linked Data on the Web Workshop (LDOW2009), CEUR Workshop Proceedings*, Madrid, Spain, 2009.
- Robert B. Allen. Retrieval from facet spaces. *Electronic Publishing*, 8:247–257, 1995.
- Samur Araujo, Daniel Schwabe, and Simone Barbosa. **Experimenting with explorer: a direct manipulation generic rdf browser and querying tool**. In *Workshop on Visual Interfaces to the Social and the Semantic Web (VISSW2009)*, Sanibel Island, Florida, USA, February 2009.
- Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. **Dbpedia: A nucleus for a web of open data**. In *Proceedings of the 6th international The Semantic Web and 2nd Asian conference on Asian Semantic Web Conference, ISWC'07/ASWC'07*, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-76297-3, 978-3-540-76297-3.
- Manoochehr Azmoodeh and Hongbo Du. **Gql, a graphical query language for semantic databases**. In Maurizio Rafanelli, John Klensin, and Per Svensson, editors, *Statistical and Scientific Database Management*, volume 339 of *Lecture Notes in Computer Science*, pages 259–277. Springer Berlin / Heidelberg, 1989. ISBN 978-3-540-50575-4. 10.1007/BFb0027518.
- Marcia J. Bates. **How to use controlled vocabularies more effectively in online searching**. *Online*, 12:45–56, November 1988. ISSN 0146-5422.
- Tim Berners-Lee. Linked Data - Design Issues. <http://www.w3.org/DesignIssues/LinkedData.html>, 2006. [Online; accessed 21-Nov-2011].
- Tim Berners-lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. **Tabulator: Exploring and analyzing linked data on the semantic web**. In *In Proceedings of the 3rd International Semantic Web User Interaction Workshop (SWUI06)*, page 06, 2006.

- Tim Berners-Lee, James Hendler, and Ora Lassila. **The semantic web: Scientific american**. *Scientific American*, 284(5):28–37, May 2001.
- Tim Berners-Lee, Jill Hollenbach, Kanghao Lu, Joe Presbrey, Eric Prud’ommeaux, and m.c. schraefel. **Tabulator redux: Writing into the semantic web**. Technical report, University of Southampton, 2007.
- Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. **Soylent: a word processor with a crowd inside**. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology*, UIST ’10, pages 313–322, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0271-5.
- Jeffrey P. Bigham, Ryan S. Kaminsky, and Jeffrey Nichols. **Mining web interactions to automatically create mash-ups**. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST ’09, pages 203–212, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-745-5.
- Chris Bizer, Anja Jentzsch, and Richard Cyganiak. **State of the LOD Cloud**. Technical report, Free University of Berlin, March 2011.
- Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sren Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. **Dbpedia - a crystallization point for the web of data**. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3): 154 – 165, 2009. ISSN 1570-8268. The Web of Data.
- Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. **Freebase: a collaboratively created graph database for structuring human knowledge**. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD ’08, pages 1247–1250, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-102-6.
- David Booth. The uri lifecycle in semantic web architecture. In *Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09)*, volume 52, page 54, 2009.
- Pia Borlund and Peter Ingwersen. The development of a method for the evaluation of interactive information retrieval systems. *Journal of documentation*, 53(3):225–250, 1997.
- Michael Bostock and Jeffrey Heer. Protovis: A graphical toolkit for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1121–1128, 2009. ISSN 1077-2626.
- Dan Brickley and Libby Miller. **FOAF Vocabulary Specification 0.97**. Namespace document, The Foaf Project, January 2010.

- Tiziana Catarci, Maria Francesca Costabile, Stefano Levialdi, and Carlo Batini. Visual query systems for databases: A survey. *J. Vis. Lang. Comput.*, 8(2):215–260, 1997.
- Pierre-Antoine Champin. **Tal4Rdf: lightweight presentation for the semantic web**. In Sören Auer, Chris Bizer, and Gunnar Aastrand Grimnes, editors, *CEUR Workshop Proceedings (2009)*, volume 449 of *CEUR Workshop Proceedings ISSN 1613-0073*, June 2009.
- Edward C. Clarkson, Shamkant B. Navathe, and James D. Foley. **Generalized formal models for faceted user interfaces**. In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries, JCDL '09*, pages 125–134, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-322-8.
- Edgar F. Codd. **A relational model of data for large shared data banks**. *Commun. ACM*, 26(1):64–69, January 1983. ISSN 0001-0782.
- Edward Cutrell, Daniel Robbins, Susan Dumais, and Raman Sarin. **Fast, flexible filtering with phlat**. In *Proceedings of the SIGCHI conference on Human Factors in computing systems, CHI '06*, pages 261–270, New York, NY, USA, 2006. ACM. ISBN 1-59593-372-7.
- Aba-Sah Dadzie and Matthew Rowe. **Approaches to visualising linked data: A survey**. *Semantic Web*, 2(2):89–124, 2011.
- Mark Derthick, John Kolojejchick, and Steven F. Roth. **An interactive visual query environment for exploring data**. In *Proceedings of the 10th annual ACM symposium on User interface software and technology, UIST '97*, pages 189–198, New York, NY, USA, 1997. ACM. ISBN 0-89791-881-9.
- Alan Dix, Janet Finlay, Gregory D. Abowd, and Russell Beale. *Human Computer Interaction*. Pearson, Harlow, England, 3. edition, 2003. ISBN 978-0-13-046109-4.
- Mira Dontcheva, Steven M. Drucker, Geraldine Wade, David Salesin, and Michael F. Cohen. **Summarizing personal web browsing sessions**. In *Proceedings of the 19th annual ACM symposium on User interface software and technology, UIST '06*, pages 115–124, New York, NY, USA, 2006. ACM. ISBN 1-59593-313-1.
- Mira Dontcheva, Sharon Lin, Steven M Drucker, and Michael F Cohen. **Experiences with content extraction from the web**. In *SIGCHI 2008 Workshop on Semantic Web User Interaction*, Florence, Italy, 2008.
- Basil Ell, Denny Vrandečić, and Elena Simperl. Spartiquation: Verbalizing sparql queries. In *Proceedings of the Interacting with Linked Data (ILD) Workshop at ESWC 2012*, Heraklion, Greece, 5 2012.
- Basil Ell, Denny Vrandečić, and Elena Simperl. **Labels in the web of data**. In *Proceedings of the 10th international conference on The semantic web - Volume Part I, ISWC'11*, pages 162–176, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-25072-9.

- Rob Ennals, Eric Brewer, Minos Garofalakis, Michael Shadle, and Prashant Gandhi. [Intel mash maker: join the web](#). *SIGMOD Rec.*, 36:27–33, December 2007. ISSN 0163-5808.
- Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007. ISBN 3540496114.
- Alexander Faaborg and Henry Lieberman. [A goal-oriented web browser](#). In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 751–760, New York, NY, USA, 2006. ACM. ISBN 1-59593-372-7.
- Thomas M. J. Fruchterman and Edward M. Reingold. [Graph drawing by force-directed placement](#). *Software: Practice and Experience*, 21(11):1129–1164, 1991. ISSN 1097-024X.
- Jonathan Gray, Lucy Chambers, and Liliana Bounegru, editors. *The Data Journalism Handbook*. Oreilly and Associates Inc, 2012. ISBN 9781449330064 1449330061. CC BY-SA 3.0.
- Thomas R. Gruber. [A translation approach to portable ontology specifications](#). *Knowl. Acquis.*, 5:199–220, June 1993. ISSN 1042-8143.
- Harry Halpin, Patrick J. Hayes, and Henry S. Thompson. When owl: sameas isn't the same redux: A preliminary theory of identity and inference on the semantic web. In *LDH*, pages 25–30, 2011.
- Andreas Harth. VisiNav: Visual Web Data Search and Navigation. In *Database and Expert Systems Applications*, pages 214–228. Springer, 2009.
- Olaf Hartig, Christian Bizer, and Johann-Christoph Freytag. [Executing sparql queries over the web of linked data](#). In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, pages 293–309, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-04929-3.
- Tuukka Hastrup, Richard Cyganiak, and Uldis Bojars. [Browsing Linked Data with Fenfire](#). In Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee, editors, *Proceedings of the Linked Data on the Web Workshop, Beijing, China*, volume 369 of *CEUR Workshop Proceedings ISSN 1613-0073*, April 2008.
- Michael Hausenblas, Wolfgang Halb, Yves Raimond, Lee Feigenbaum, and Danny Ayers. [Scovo: Using statistics on the web of data](#). In Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Paslaru Bontas Simperl, editors, *ESWC*, volume 5554 of *Lecture Notes in Computer Science*, pages 708–722. Springer, 2009. ISBN 978-3-642-02120-6.

- Bin He, Mitesh Patel, Zhen Zhang, and Kevin Chen-Chuan Chang. **Accessing the deep web**. *Commun. ACM*, 50:94–101, May 2007. ISSN 0001-0782.
- Marti A. Hearst. Next generation web search: Setting our sites. *IEEE Data Eng. Bull.*, 23(3):38–48, 2000.
- Marti A. Hearst. Design recommendations for hierarchical faceted search interfaces. In *SIGIR, Workshop on Faceted Search*, Seattle, WA, USA, 2006.
- Marti A. Hearst. *Search User Interfaces*. Cambridge University Press, New York, NY, USA, 1st edition, 2009. ISBN 0521113792, 9780521113793.
- Marti A. Hearst, Ame Elliott, Jennifer English, Rashmi Sinha, Kirsten Swearingen, and Ka-Ping Yee. **Finding the flow in web site search**. *Commun. ACM*, 45:42–49, September 2002. ISSN 0001-0782.
- Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011.
- Jeffrey Heer and Danah Boyd. **Vizster: Visualizing online social networks**. In *Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, pages 5–, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7803-9464-x.
- Philipp Heim, Thomas Ertl, and Jrgen Ziegler. **Facet graphs: Complex semantic querying made easy**. In *Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010)*, volume 6088 of *LNCIS*, pages 288–302, Berlin/Heidelberg, 2010. Springer. ISBN 978-3-642-13485-2.
- Philipp Heim, Steffen Lohmann, Davaadorj Tsendragchaa, and Thomas Ertl. **Semlens: Visual analysis of semantic data with scatter plots and semantic lenses**. In *Proceedings of the 7th International Conference on Semantic Systems (I-SEMANTICS 2011)*, pages 175–178, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0621-8.
- Philipp Heim, Jürgen Ziegler, and Steffen Lohmann. **gFacet: A browser for the web of data**. In Sören Auer, Sebastian Dietzold, Steffen Lohmann, and Jürgen Ziegler, editors, *Proceedings of the International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW’08)*, pages 49–58. CEUR-WS, 2008.
- Michiel Hildebrand and Jacco van Ossenbruggen. **Configuring Semantic Web Interfaces by Data Mapping**. In Siegfried Handschuh, Tom Heath, and Vinhtuan Thai, editors, *Visual Interfaces to the Social and the Semantic Web (VISSW 2009)*, volume 443, February 2009.
- Michiel Hildebrand, Jacco van Ossenbruggen, and Lynda Hardman. */facet: A browser for heterogeneous semantic web repositories*. In *International Semantic Web Conference*, pages 272–285, 2006.

- Aidan Hogan, Andreas Harth, Alexandre Passant, Stefan Decker, and Axel Polleres. Weaving the pedantic web. In *LDOW*, 2010.
- John Howse, Gem Stapleton, Kerry Taylor, and Peter Chapman. **Visualizing ontologies: A case study**. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist, editors, *International Semantic Web Conference (1)*, volume 7031 of *Lecture Notes in Computer Science*, pages 257–272. Springer, 2011. ISBN 978-3-642-25072-9.
- Scott. E. Hudson. User interface specification using an enhanced spreadsheet model. *ACM Transactions on Graphics (TOG)*, 13(3):239, 1994.
- David Huynh, Stefano Mazzocchi, and David Karger. **Piggy bank: Experience the semantic web inside your web browser**. *Web Semant.*, 5:16–27, March 2007a. ISSN 1570-8268.
- David F Huynh and David R Karger. Adopting a common data model for end-user web programming tools, 2009a.
- David F. Huynh and David R. Karger. Parallax and companion: Set-based browsing for the data web. In *WWW Conference*. ACM, 2009b.
- David F. Huynh, David R. Karger, and Robert C. Miller. **Exhibit: lightweight structured data publishing**. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 737–746, New York, NY, USA, 2007b. ACM. ISBN 978-1-59593-654-7.
- David F. Huynh, David R. Karger, and Robert C. Miller. **Exhibit: lightweight structured data publishing**. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 737–746, New York, NY, USA, 2007c. ACM. ISBN 978-1-59593-654-7.
- David F. Huynh, Robert C. Miller, and David R. Karger. **Enabling web browsers to augment web sites' filtering and sorting functionalities**. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, UIST '06, pages 125–134, New York, NY, USA, 2006. ACM. ISBN 1-59593-313-1.
- David F. Huynh, Robert C. Miller, and David R. Karger. Potluck: Data mash-up tool for casual users. *Web Semant.*, 6(4):274–282, 2008. ISSN 1570-8268.
- Mustafa Jarrar and Marios D. Dikaiakos. Mashql: a query-by-diagram topping sparql. In *ONISW '08: Proceeding of the 2nd international workshop on Ontologies and nformation systems for the semantic web*, pages 89–96, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-255-9.
- David R. Karger, Scott Ostler, and Ryan Lee. The web page as a wysiwyg end-user customizable database-backed information management application. In *UIST '09*, pages 257–260, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-745-5.

- Esther Kaufmann and Abraham Bernstein. How useful are natural language interfaces to the semantic web for casual end-users? In *ISWC'07/ASWC'07: Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, pages 281–294, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-76297-3, 978-3-540-76297-3.
- Esther Kaufmann and Abraham Bernstein. **Evaluating the usability of natural language query languages and interfaces to semantic web knowledge bases**. *Web Semant.*, 8: 377–393, November 2010. ISSN 1570-8268.
- Aniket Kittur, Ed H. Chi, and Bongwon Suh. **Crowdsourcing user studies with mechanical turk**. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 453–456, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1.
- Gary Klein, Jennifer K. Phillips, Erica L. Rall, and Deborah A. Peluso. **A Data/Frame Theory of Sense Making**. In *Expertise out of context: proceedings of the sixth International Conference on Naturalistic Decision Making*, pages 113–155, 2003.
- Georgi Kobilarov and Ian Dickinson. Humboldt: Exploring linked data. In Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee, editors, *LDOW*, volume 369 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
- Günter Ladwig and Thanh Tran. **Linked data query processing strategies**. In *Proceedings of the 9th international semantic web conference on The semantic web - Volume Part I*, ISWC'10, pages 453–469, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-17745-X, 978-3-642-17745-3.
- Danh Le-Phuoc, Axel Polleres, Manfred Hauswirth, Giovanni Tummarello, and Christian Morbidoni. **Rapid prototyping of semantic mash-ups through semantic web pipes**. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 581–590, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-487-4.
- Kristina Lerman, Lise Getoor, Steven Minton, and Craig Knoblock. **Using the structure of web sites for automatic segmentation of tables**. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 119–130, New York, NY, USA, 2004. ACM. ISBN 1-58113-859-8.
- Steffen Lohmann, Philipp Heim, Timo Stegemann, and Jrgen Ziegler. **The relfinder user interface: Interactive exploration of relationships between objects of interest**. In *Proceedings of the 14th International Conference on Intelligent User Interfaces (IUI 2010)*, pages 421–422, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-515-4.
- Gary Marchionini and Ryen White. **Find what you need, understand what you find**. *International Journal of Human-Computer Interaction*, 23(3):205–237, 2008.

- m.c. schraefel and David Karger. **The pathetic fallacy of rdf**. In *International Workshop on the Semantic Web and User Interaction (SWUI) 2006*, 2006.
- Deborah L. McGuinness. **Ontologies come of age**. In Dieter Fensel, Jim Hendler, Henry Lieberman, and Wolfgang Wahlster, editors, *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, 2003.
- Christian Morbidoni, Axel Polleres, and Giovanni Tummarello. **Who the foaf knows alice? a needed step towards semantic web pipes**. In *ISWC 2007 Workshop on New forms of Reasoning for the Semantic Web: Scaleable, Tolerant and Dynamic*, November 2007.
- Enrico Motta, Paul Mulholland, Silvio Peroni, Mathieu d’Aquin, Jos Manuel Gmez-Prez, Victor Mendez, and Fouad Zablith. **A novel approach to visualizing and navigating ontologies**. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist, editors, *International Semantic Web Conference (1)*, volume 7031 of *Lecture Notes in Computer Science*, pages 470–486. Springer, 2011. ISBN 978-3-642-25072-9.
- Bonnie A. Nardi and James R. Miller. An ethnographic study of distributed problem solving in spreadsheet development. In *CSCW ’90: Proceedings of the 1990 ACM conference on Computer-supported cooperative work*, pages 197–208, New York, NY, USA, 1990. ACM. ISBN 0-89791-402-3.
- Chris North and Ben Shneiderman. **Snap-together visualization: a user interface for coordinating visualizations via relational schemata**. In *Proceedings of the working conference on Advanced visual interfaces*, AVI ’00, pages 128–135, New York, NY, USA, 2000. ACM. ISBN 1-58113-252-2.
- Benjamin Nowack. **Paggr: Linked data widgets and dashboards**. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(4):272 – 277, 2009. ISSN 1570-8268. Semantic Web challenge 2008.
- Tope Omitola, Christos L. Koumenides, Igor O. Popov, Yang Yang, Manuel Salvadores, Martin Szomszor, Tim Berners-Lee, Nicholas Gibbins, Wendy Hall, m. c. schraefel, and Nigel Shadbolt. Put in your postcode, out comes the data: A case study. In *ESWC (1)*, pages 318–332, 2010.
- Eyal Oren, Renaud Delbru, and Stefan Decker. **Extending faceted navigation for rdf data**. In *Proceedings of the 5th international conference on The Semantic Web*, ISWC’06, pages 559–572, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-49029-9, 978-3-540-49029-6.
- Adam Perer and Ben Shneiderman. **Systematic yet flexible discovery: guiding domain experts through exploratory data analysis**. In *Proceedings of the 13th international*

- conference on Intelligent user interfaces*, IUI '08, pages 109–118, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-987-6.
- Emmanuel Pietriga, Christian Bizer, David Karger, and Ryan Lee. Fresnel - a browser-independent presentation vocabulary for rdf. In *In: Proceedings of the Second International Workshop on Interaction Design and the Semantic Web*, pages 158–171. Springer, 2006.
- Peter Pirolli and Stuart Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of International Conference on Intelligence Analysis*, volume 5, pages 2–4, 2005.
- Catherine Plaisant, Ben Shneiderman, Khoa Doan, and Tom Bruns. **Interface and data architecture for query preview in networked information systems**. *ACM Trans. Inf. Syst.*, 17:320–341, July 1999. ISSN 1046-8188.
- Annabel Pollock and Andrew Hockley. What’s wrong with internet searching. *D-Lib Magazine*, March 1997. ISSN 1082-9873.
- Eric Prudhommeaux and Andy Seaborne. **Sparql query language for rdf**. W3C Recommendation, January 2008.
- Dennis. A. Quan and David R. Karger. **How to make a semantic web browser**. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 255–265, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X.
- Alexander J. Quinn and Benjamin B. Bederson. **Human computation: a survey and taxonomy of a growing field**. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1403–1412, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9.
- Sarada. R. Ranganathan. *Colon Classification: Basic Classification*. Ess Ess Publication, 1933.
- Davi De Castro Reis, Reis Paulo, Alberto H.F. Laender, Paulo B. Golgher, and Altigran S. da Silva. **Automatic web news extraction using tree edit distance**. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 502–511, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X.
- Mary Beth Rosson and John M. Carroll. **Scenario-based design**. In Julie A. Jacko and Andrew Sears, editors, *The human-computer interaction handbook*, pages 1032–1050. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 2003. ISBN 0-8058-3838-4.
- Daniel M Russell, Robin Jeffries, and Lilly Irani. Sensemaking for the rest of us. In *CHI 2008 Sensemaking workshop*, 2008.

- Daniel M. Russell, Mark J. Stefik, Peter Pirolli, and Stuart K. Card. **The cost structure of sensemaking**. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, CHI '93, pages 269–276, New York, NY, USA, 1993. ACM. ISBN 0-89791-575-5.
- Cristina Sarasua, Elena Simperl, and Natalya Fridman Noy. Crowdfmap: Crowdsourcing ontology alignment with microtasks. In *International Semantic Web Conference (1)*, pages 525–541, 2012.
- 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. ISBN 1-59593-168-6.
- M. C. schraefel, Yuxiang Zhu, David Modjeska, Daniel Wigdor, and Shengdong Zhao. **Hunter gatherer: interaction support for the creation and management of within-web-page collections**. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, pages 172–181, New York, NY, USA, 2002. ACM. ISBN 1-58113-449-5.
- Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt. **Fedx: optimization techniques for federated query processing on linked data**. In *Proceedings of the 10th international conference on The semantic web - Volume Part I*, ISWC'11, pages 601–616, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-25072-9.
- Abigail J. Sellen, Rachel Murphy, and Kate L. Shaw. **How knowledge workers use the web**. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 227–234, New York, NY, USA, 2002. ACM. ISBN 1-58113-453-3.
- Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. **The semantic web revisited**. *IEEE Intelligent Systems*, 21:96–101, May 2006. ISSN 1541-1672.
- Nigel R. Shadbolt, Nicholas Gibbins, Hugh Glaser, Stephen Harris, and m.c. schraefel. **Cs aktive space or how we stopped worrying and learned to love the semantic web**. *IEEE Intelligent Systems*, 19(3):41–47, 2004.
- Yuan Kui Shen and David R. Karger. **U-rest: an unsupervised record extraction system**. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 1347–1348, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7.
- Ben Shneiderman. **Direct manipulation: A step beyond programming languages**. *Computer*, 16(8):57–69, August 1983. ISSN 0018-9162.

- Ben Shneiderman. **The eyes have it: A task by data type taxonomy for information visualizations**. In *Proceedings of the 1996 IEEE Symposium on Visual Languages, VL '96*, pages 336–, Washington, DC, USA, 1996. IEEE Computer Society. ISBN 0-8186-7508-X.
- Daniel Alexander Smith and mc schraefel. **Interactively using semantic web knowledge: Creating scalable abstractions with facetontology**. Technical report, University of Southampton, 2008.
- Daniel Alexander Smith, Igor Popov, and mc schraefel. **Data picking linked data: Enabling users to create faceted browsers**. In *Web Science Conference 2010*, March 2010.
- Moritz Stefaner and Boris Muller. **Elastic lists for facet browsers**. In *Proceedings of the 18th International Conference on Database and Expert Systems Applications*, pages 217–221, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2932-1.
- Egemen Tanin, Richard Beigel, and Ben Shneiderman. **Design and evaluation of incremental data structures and algorithms for dynamic query interfaces**. In *Proceedings of the 1997 IEEE Symposium on Information Visualization (InfoVis '97)*, INFOVIS '97, pages 81–, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-8189-6.
- Maryam Tohidi, William Buxton, Ronald Baecker, and Abigail Sellen. **Getting the right design and the design right**. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '06*, pages 1243–1252, New York, NY, USA, 2006. ACM. ISBN 1-59593-372-7.
- Michael Toomim, Steven M. Drucker, Mira Dontcheva, Ali Rahimi, Blake Thomson, and James A. Landay. **Attaching ui enhancements to websites with end users**. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '09*, pages 1859–1868, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7.
- Giovanni Tummarello, Renaud Delbru, and Eyal Oren. **Sindice.com: weaving the open linked data**. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference, ISWC'07/ASWC'07*, pages 552–565, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-76297-3, 978-3-540-76297-3.
- Fernanda B. Viegas, Martin Wattenberg, Frank van Ham, Jesse Kriss, and Matt McKeon. Manyeyes: a site for visualization at internet scale. *IEEE Transactions on Visualization and Computer Graphics*, 13:1121–1128, 2007. ISSN 1077-2626.
- Denny Vrandečić. **Wikidata: a new platform for collaborative data collection**. In *Proceedings of the 21st international conference companion on World Wide Web, WWW*

- '12 Companion, pages 1063–1064, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1230-1.
- Wesley Willett, Jeffrey Heer, and Maneesh Agrawala. **Scented widgets: Improving navigation cues with embedded visualizations**. *IEEE Transactions on Visualization and Computer Graphics*, 13:1129–1136, November 2007. ISSN 1077-2626.
- Max L. Wilson, Paul Andre, and m.c. schraefel. **Backward highlighting: Enhancing faceted search**. In *Proceedings of the 21st Symposium on User Interface Software and Technology (UIST2008)*, pages 235–238, April 2008.
- Max L. Wilson, m.c. schraefel, and Ryen W. White. **Evaluating advanced search interfaces using established information-seeking models**. *Journal of the American Society for Information Science and Technology*, 60(7):1407–1422, 2009. ISSN 1532-2890.
- Jeffrey Wong and Jason I. Hong. **Making mashups with marmite: towards end-user programming for the web**. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 1435–1444, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-593-9.
- Miao Xiong, YiFan Chen, Hao Zheng, and Yong Yu. **Towards quick understanding and analysis of large-scale ontologies**. In Riichiro Mizoguchi, Shi Zhongzhi, and Giunchiglia Fausto, editors, *The Semantic Web ASWC 2006*, volume 4185 of *Lecture Notes in Computer Science*, pages 84–98. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-38329-1.
- Yang Yang, Priyanka Singh, Jiadi Yao, Ching-man Au Yeung, Amir Zareian, Xiaowei Wang, Zhonglun Cai, Manuel Salvadores, Nicholas Gibbins, Wendy Hall, and Nigel Shadbolt. **Distributed human computation framework for linked data co-reference resolution**. In *Proceedings of the 8th extended semantic web conference on The semantic web: research and applications - Volume Part I*, ESWC'11, pages 32–46, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-21033-4.
- Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti A. Hearst. **Faceted metadata for image search and browsing**. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '03, pages 401–408, New York, NY, USA, 2003. ACM. ISBN 1-58113-630-7.
- Yanhong Zhai and Bing Liu. **Web data extraction based on partial tree alignment**. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 76–85, New York, NY, USA, 2005. ACM. ISBN 1-59593-046-9.
- Junliang Zhang and Gary Marchionini. **Evaluation and evolution of a browse and search interface: Relation browser++**. In *Proceedings of the 2005 national conference on Digital government research*, dg.o '05, pages 179–188. Digital Government Society of North America, 2005.

- Xingjian Zhang and Jeff Heflin. Using tag clouds to quickly discover patterns in linked data sets. In Olaf Hartig, Andreas Harth, and Juan Sequeda, editors, *COLD*, volume 782 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.
- M. Moshe Zloof. **Query-by-example: a data base language**. *IBM Syst. J.*, 16:324–343, December 1977. ISSN 0018-8670.