ARCHITECTURE AND METHODOLOGIES FOR ADAPTIVE PERSONALISATION ON THE WEB OF DATA

Benjamin Heitmann

DERI TECHNICAL REPORT 2010-02-15
FEBRUARY 2010
Abstract. Personalisation through recommendation plays an important role for the user experience of many e-commerce and social media sites. Existing services have accumulated the data which is required for good recommendations over time. In order to compete with existing services, new service providers need acquire data and knowledge in order to provide relevant recommendations. We propose to use the Web of Data for the purpose of recommendation and personalisation. In this report we present a survey of the state of the art in the area of adaptive personalisation. The four main research challenges in relation to the design of adaptive services are: the acquisition of (1) data and (2) knowledge, the selection of relevant features from an open corpus (3), and the scalability (4) of the architecture, algorithms and data structures used.

This report describes our contributions towards addressing these research challenges: (i) The implementation and evaluation of a collaborative filtering algorithm in the music domain, which uses the Web of Data to address the data acquisition problem. (ii) The description of a use case, of a detailed example and of available Linked Data data sources to address the knowledge acquisition problem. (iii) A detailed example and a discussion of possible approaches to address the open corpus problem on the Web of Data by using feature selection and soft case-based reasoning. (iv) An architecture for open and scalable recommender systems on the Web of Data which is derived from our reference architecture for Semantic Web applications and based on an empirical analysis of Semantic Web applications.

We also show that the Web of Data is not only well suited as data source for current recommendation algorithms, it also allows us to go beyond the capabilities of current approaches for adaptive personalisation and recommendation.

Keywords: adaptive personalisation, recommender systems, architecture, Semantic Web, Web of Data, scalability, openness, data acquisition, knowledge acquisition, open corpus

1DERI (Digital Enterprise Research Institute), National University of Ireland, Galway, IDA Business Park, Lower Dangan, Galway, Ireland. E-mail: benjamin.heitmann@deri.org.

Acknowledgements: The work presented in this paper has been funded by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Líon-2). This work has greatly benefited from the feedback of Conor Hayes in his role as Ph.D. supervisor, and discussion and comments of Richard Cyganiak, Stefan Decker, Laura Drăgan, Michael Hausenblas, Pascal Jürgens, Marcel Karnstedt, Sheila Kinsella, Knud Möller, Alexandre Passant, Jürgen Umbrich.

Submitted as first year Ph.D. transfer report

Copyright © 2010 by the authors
1 Introduction

Personalised recommendations have proven themselves to greatly enhance the user experience of searching, exploring and finding new and interesting content and products\[50\]. Examples include some of the most well known e-commerce web sites\[62\] such as Amazon\[50\] and popular social media sites\[31\] for discovering movies (Netflix\[140\]) or music (Last.fm\[229\]).

However, in order to provide an attractive and successful recommendation service, appropriate data and knowledge is required, depending on the domain of the service and the algorithm used\[4\]. The most prominent personalised services have acquired the necessary data over the time since they started operating. The result is a high entry barrier for new service providers, as they have to compete with services which have had a head start in terms of the already collected data.

While most recommender systems collect data from their own users or from the content which they offer, an alternative approach is to use openly available data and content\[17\]. This is reflected in a classification of recommendation systems into two groups, based on their data sources \[17\] and their architecture:

Closed corpus recommenders only use data collected in-house for the purpose of providing recommendations for a specific domain and scenario. They are based on the assumption of a closed architecture, where all data sources are known in advance.

Open corpus recommenders provide recommendations using public data sources which are relevant for the recommendation domain and scenario. They require a new type of open architecture, which must take discovery, selection and integration of data sources without prior knowledge into account. The different steps of the recommendation process can be performed by different parties.

In this report, we use the notion of software architecture as an abstraction of a runtime system, which specifies the roles of the different parts of the system and how they interact \[27\]. Software architecture can be used to address cross-cutting concerns, such as scalability and openness, which can not be addressed by just changing one part of a system \[2\]. We provide our definition of software architecture in section \[8.1\] and explain how it applies to the Web and the Web of Data.

Based on a literature survey, we have identified the four main research challenges related to the architectural design of recommendation and personalisation services and their use of an open corpus:

1. Most recommendation algorithms require sufficient amounts of data in order to provide relevant recommendations. This makes it a challenge to provide recommendations for new users, new items or sparse data sets. This is known as the data acquisition problem \[4\] \[19\] \[63\] \[6\]

2. For many recommendation scenarios domain knowledge is required in addition to data about users and items, however knowledge engineering is expensive, which characterises the knowledge acquisition problem \[19\] \[4\] \[68\]

3. Both data and knowledge can be provided by open data sources, however in order to use open data the sources and data subsets with the most relevant features amongst the available sources have to be selected. This characterises the open corpus problem \[17\] \[3\]

http://amazon.com
http://netflix.com/
http://last.fm
4. Lastly, if data sets with relevant data or knowledge have been selected, then the recommendation algorithms and data structures need to be scalable for the amount of collected data and knowledge.

In order to address these challenges, we propose to use the Web of Data as an open data source. In recent years the Web has developed into a global information space consisting not just of linked documents but also of linked data, which has resulted in the emergence of the Web of Data. An increasing number of data providers from many different domains are contributing to it and the number of sources is growing by the month.

In order to address the identified research challenges, the Web of Data provides structured data and ontologies from different domains and communities. These data providers can be used as sources for the data and knowledge which is required by recommendation algorithms. Semantic annotations are part of the Web of Data, and provide the means for selecting the most relevant individual data sources or subsets of a source.

However using the Web of Data for recommender systems introduces new scalability requirements for both the architecture and the algorithms used. We will describe a software architecture which allows recommender systems to be open and scalable. This architecture is motivated from the similarities between open recommenders and open applications on the Web of Data. We will explain how to use components from a reference architecture for Semantic Web applications in order to build open and scalable recommender systems on the Web of Data.

The Web of Data also provides sources which contain data from different domains or which link users from different communities. These intrinsic links can be followed between data sources, thus providing connections between related data from different domains and identical users in different communities. Data which connects related topics and users on such a scale has not been available through public data sources before, and as such provides both new opportunities as well as new challenges for the next generation of adaptive personalisation.

1.1 Research goals

The goal of this PhD research is to show, that the Web of Data is not only well suited as data source for current recommendation algorithms, it also allows us to go beyond the capabilities of current approaches for adaptive personalisation and recommendation. We will demonstrate the viability of using the Web of Data as an open corpus, by implementing and evaluating prototypes, which show that suitable data for established recommendation algorithms is available. In addition, we will explore new approaches for exploiting the intrinsic links between data sources, in order to provide adaptive personalisation services which go beyond the state of the art.

1.2 Research contributions

In the current state of the PhD research, the following contributions have been made:

1. the identification of the main challenges related to the architectural design of recommendation and personalisation services and their use of an open corpus, based on a literature survey of the state of the art (Section 3)

2. an implementation and evaluation of a collaborative filtering algorithm in the music domain, which uses the Web of Data to address the data acquisition problem (Section 7)
3. the description of a use case (Section 4), of a detailed example (Section 5.1) and of available Linked Data data sources (Section 6.2) to address the knowledge acquisition problem

4. a detailed example and a discussion of possible approaches (both in Section 5) to address the open corpus problem on the Web of Data by using feature selection and case-based reasoning

5. an architecture for open and scalable recommender systems on the Web of Data which is derived from a reference architecture for Semantic Web applications. The reference architecture is based on an empirical analysis of 98 Semantic Web applications (Section 8). An Example of applying this architecture to a recommender system is given.

1.3 Outline

This report is structured as follows: In order to provide a survey of the state of the art in the area of adaptive personalisation and recommendation, Section 2 describes the field of adaptive personalisation. We present both established recommendation algorithms as well as emerging approaches for harnessing the collective intelligence of a community are presented.

Based on this, in Section 3 we introduce the four main challenges related to the architectural design of recommendation and personalisation services and their use of an open corpus. Related literature for each challenge is provided.

We then describe the principles of the Web of Data and the types of available data sources in Section 4. This shows the feasibility of using the Web of Data for addressing the data and knowledge acquisition problems.

Section 5 describes how the intrinsic links between different data sources on the Web of Data can be exploited to go beyond the capabilities of the current generation of adaptive personalisation. We use a detailed example to describe the benefits of following links between different data sources, and then discuss the usage of feature selection and case-based reasoning to address the open corpus problem.

Section 6 builds on the sections about the Web of Data and about the intrinsic features of Linked Data, and introduces three uses cases which illustrate the end-user benefits of using the Web of Data for adaptive personalisation. Use cases are described for using collaborative filtering to recommend music, using case-based reasoning for expert finding, and providing a context based browsing agent. Section 7 builds on one of these use cases, and describes the implementation and evaluation of a collaborative filtering algorithm for the music domain in order to address the data acquisition problem.

Section 8 addresses the scalability of recommender systems which use the Web of Data as an open corpus by specifying an architecture for such open and scalable recommender systems. The reference architecture is based on an empirical analysis of 98 Semantic Web applications. An example of applying this architecture to a recommender system is given.

analysing the architectural requirements for making Semantic Web applications scalable. The chapter is based on an empirical analysis of 98 Semantic Web applications.

Finally, we conclude this report by providing a summary of the results at this stage of the PhD research, and list the future research agenda as an appendix.
2 Background: Adaptive personalisation and recommender systems

Providing recommendations is a popular way of providing adaptive personalisation on e-commerce and Web 2.0 sites. The algorithms which make recommendation services possible have data requirements which are specific to the domain and the scenario of the recommendation. While most recommender systems use their own closed corpus of data, an alternative is to use publicly available data sources as an open corpus for the recommendation algorithm. This lowers the entry barriers for new service providers, but it also introduces new challenges.

In this chapter we introduce different approaches for adaptive personalisation systems. First we describe the class of adaptive web systems in general. Then we list the main classes of recommendation algorithms and their data requirements, according to established recommender systems literature. As a comparison we outline recent, more lightweight approaches to provide personalisation based on collective intelligence.

2.1 Adaptive personalisation

Adaptive Web systems belong to the class of user-adaptive software systems [64]. Adaptive systems use explicit user models to represent user knowledge, goals and interests. These user models are then used by an adaptive web system to provide a personalised experience for different users, whereas adaptable systems must be explicitly changed by the user, e.g. by rearranging elements on a web site [18].

Personalisation techniques for adaptive web systems can be grouped in three areas [17]: personalisation of information retrieval, personalisation of browsing and personalisation through filtering and recommendation. Web systems for adaptive recommendation are a subtype of adaptive web systems, which attempt to deduce the users goals and interests from his browsing activity, and build a list of relevant links which are recommended to the user [16]. We are going to focus on using filtering and recommendations for adaptive personalisation, because interpreting the users actions in terms of his goals allows us to have a bigger impact on his user experience [62], than just building a database of activity data.

2.2 Recommender systems

Personalised recommendations have proven themselves to greatly enhance the user experience of searching, exploring and finding new and interesting content [50]. In the domain of e-commerce, recommender systems have the potential to support and improve the quality of customer decisions, by reducing the information overload facing consumers as well as the complexity of online searches [71]. In addition, e-commerce web sites benefit from recommender systems by converting browsers into buyers, increasing the average order size and improving consumer loyalty [62].

Recommender systems require three components to provide recommendations [19]: (1) **background data**, which is the information the system has before the recommendation process begins, (2) **input data**, which is the information provided about the user in order to make a recommendation, and (3) the **recommender algorithm** which operates on background and input data in order to provide recommendations for a user.

Different recommendation algorithms require different types of background and input data in order to provide recommendations. Current recommendation algorithms can be grouped in 4 classes [19, 4]:

(i) **Collaborative filtering** aggregates ratings for items from different users, and uses similarities between items to recommend items. It is probably the most mature and widely implemented recommendation algorithm, because it achieves fairly good results and is easy to implement. It only requires data about the ratings between users and items as background data, no other information about either the users or the items
is required. The input data usually consists of a user profile providing ratings for one or more items. The recommendation algorithm uses the background data to calculate the pair-wise similarity between all items or all users, and then uses the input data to recommend similar users or items.

(ii) **Content-based recommendation** uses the features of the items as the background data for the recommendation. These can either be directly derived from the content, e.g. keywords from text, tempo of music, or derived from the meta-data of the items, e.g. author, title and genre. The input data needs to describe the users preferences in terms of content features. Both the background and input data requires the consistent description of content features, in order to match the user preferences to the features of the content.

(iii) **Knowledge-based recommendation** aims to suggest items based on inferences about a users needs and preferences. This requires background data which includes knowledge about users and items, which is sufficient in consistency and scale for making inferences. The input data needs to provide knowledge about the users needs and preferences which can be mapped to the knowledge about users and items in the background data.

(iv) **Hybrid algorithms** combine two or more recommendation algorithms to provide better results with fewer of the drawbacks of an individual algorithm. In order to combine two algorithms different methods can be used [19], e.g.: the scores of several algorithms can be combined with weights; the output of one algorithm can be used as the input for the next one, thus forming a cascade; the system can switch between different algorithms depending on the situation; the presentation of the output of several algorithms can be mixed in the user interface. Most commonly, collaborative filtering is combined with some other algorithm, e.g. a content-based one, in order to mitigate situations in which not enough background data for an item or a user is available.

To summarise the data required by the different classes of recommendation algorithms: both content and knowledge-based recommendation require high quality background data about users and items. In addition, knowledge-based recommendation additionally requires domain knowledge. In contrast, the collaborative filtering algorithm only requires vectors of user-item connections. These can include numerical ratings, but can also operate on simple binary user-item connections. The requirements of hybrid algorithms are defined by the individual algorithms used as part of the hybrid.

2.3 **Collective intelligence for personalisation**

The initial categorisation of recommender systems is along the type of recommendation algorithm used. However in recent years approaches for harnessing the collective intelligence of a user community have been developed [31], which do not fit into that categorisation. Such approaches include folksonomies, object centred sociality and social network based recommendations, as well as portable identities and interchange formats for the social graph of a user.

Many Web 2.0 sites such as flickr and delicious allow users to tag content with keywords. The most popular tags are displayed on flickr to encourage users to re-use existing tags for new pictures. Delicious recommends popular tags for new bookmarks of a specific link or inside of a domain. This emergence of a collective consensus on the usage of tags is known as a folksonomy [46]. It allows personalising the user experience based on content using similar tags.

Social networking web sites provide filtering and recommendation of users or content based on both direct and in-direct connections between users. Persons are not only directly connected to other persons, but

[http://flickr.com](http://flickr.com)
[http://delicious.com](http://delicious.com)
also indirectly via objects of a social focus. In this way a community is connected to each other not only via direct links from person to person, but also via their links to e.g. music from an artist. Such communities of object centred sociality \[13\] can be used to enable better filtering of e.g. the activity stream on facebook.

Then there are recommendation services, which combine collaborative filtering algorithms with social networking elements, such as Last.fm. This allows a user to listen to recommendations based on artist similarities, e.g. artists which are similar to Johnny Cash. But users can also listen to artists which are popular with their friends or they can listen to artists which haven been tagged with the same keyword, e.g. “country”.

While collective intelligence does not require the use of traditional recommendation algorithms, it can make use of some data mining algorithms which are also used as part of recommendation services. Such algorithms include \[49\] clustering, association rule discovery, sequential pattern mining, Markov models and models with latent variables.

Combining the benefits of recommendation algorithms with the user experience of social networking will continue to play a big role in the future. Standards for reusing a user identity to log on a third-party site and reusing the social graph \[12\] of connections to friends and objects are currently emerging. Google has developed the OpenSocial API \[48\] for this purpose, while Facebook has developed Facebook Connect\[6\]. Both Google and Facebook work with many other parties on future standards for decentralised storage and access of the social graph in the DataPortability Project\[7\].

Making the portability of a users’ social graph independent of the originating site, will allow providing adaptive personalisation on a complete view of the users data. Recommendation services will not be limited to the network of friends the user has on one site, instead they will be able to use a network of friends which is spanning all sites which provide such portable social data, and on which the user has an account.

3 The main challenges for open corpus adaptive personalisation

Adaptive recommendation systems are grouped into closed corpus systems and open corpus recommender systems \[17\]. While closed corpus recommenders only recommend items from one source, open corpus recommender systems provide recommendations from a subset of the web, which is relevant to the recommendation domain and scenario.

In terms of research, closed corpus recommender systems are very well understood, while open corpus recommender systems are still an active ongoing research topic. We have identified four research challenges, related to the architectural design of recommender systems and the use of an open corpus for recommendations:

(i) Public data can be used to mitigate the problem of collecting sufficient amounts of data for the recommendation algorithm, which is known as the data acquisition problem. (ii) For many recommendation scenarios domain knowledge is required in addition to data about users and items, however knowledge engineering is expensive, which characterises the knowledge acquisition problem. (iii) Both data and knowledge can be provided by open data sources, however in order to use open data the sources and data subsets with the most relevant features amongst the available sources have to be selected. This characterises the open corpus problem. An open corpus can provide data combining different domains and different communities, which can make the recommendation more flexible regarding the domain and recommendation

---

Figure 1: A comparison of a closed and an open recommender system

request. (iv) Finally, if data sets with relevant data or knowledge have been selected, then the recommendation algorithms and data structures need to be scalable for the amount of collected data and knowledge.

Figure 1 illustrates the difference for the architecture of closed and open recommenders, while figure 2 summarises the main properties of closed and open recommender systems. In the following we will describe each of the four challenges in more detail.

3.1 The data acquisition problem

The entry barriers in terms of data acquisition are very high, which makes it hard for new service providers to compete with existing recommendation services in a domain. This can be characterised by three challenges: providing recommendations for (a) new items or for (b) new users is a challenge if no data about the item or user is available at all. Together, the new-item and new-user problems are known as the ramp-up or cold-start problem [63]. If the number of ratings is low compared to the number of items, then the (c) sparsity of the data will lead to ineffective recommendations [6]. The common cause of these challenges is the data acquisition problem: in order to provide recommendations, suitable data about the users and items needs to be available for the recommendation algorithm.

As an example, consider an online shop for digital music downloads. If the shop has music made by 100 artists, and if 200 users have rated two or more artists, then each artist has an average of 4 ratings. However there will be a power law distribution of ratings, so that there is a long tail of artists for which there are almost no ratings. However if user ratings increase by an order of magnitude to 20 ratings per user, then an average of 40 ratings per artists could provide notably better recommendations, according to our previous work.

The background of using structured data from the Web of Data to mitigate the data acquisition problem is described in section 4. We describe the implementation and evaluation of a prototype to use Linked Data for collaborative recommendations on music in section 7.
3.2 The knowledge acquisition problem

Most real-world recommender systems employ collaborative filtering or combine it with another recommendation approach, such as content based recommendation or knowledge based recommendation [4]. Such hybrid recommender systems are better suited for collections of items, such as news articles, where new items are constantly and automatically added [19]. In order to apply content based recommendation approaches, consistent metadata annotations are required to e.g. train a content classifier, which emphasises the data acquisition problem. For knowledge based recommendation approaches knowledge bases fitting the recommendation scenario are required.

This adds the knowledge acquisition problem [19], which is characterised by the high effort of knowledge engineering [68]: In order to provide recommendations in knowledge intensive recommendation scenarios, a knowledge base about the recommendation domain and the users needs to be acquired. This raises the entry barriers to providing knowledge based recommendations compared to other personalisation and adaptation approaches.

The most prominent example for the high cost of acquiring knowledge is the Cyc project, which aims to collect all of the common sense knowledge of an average human. Initially planned for 10 years and 50 million dollars, Cyc has now been in development for two decades without being finished [70].

The background of using the Web of Data as a source of formally expressed knowledge for mitigating the knowledge acquisition problem is discussed in section 4. We describe a use case which illustrates how such knowledge can be used for recommending experts for a certain topic as part of our use cases in section 6.2.

3.3 The open corpus problem

Closed corpus recommender systems only have one data source available, which usually only contains users from one community and items from one domain. This decreases the flexibility of the system regarding the recommendation scenario and decreases the ability to make recommendations across domains. In order to make the recommendation process more flexible and universal, the adaptive recommender can use an open corpus which provides knowledge from different sources [3].

Combining data and knowledge from different sources can provide connections between different do-
mains and can allow identifying identical users from different communities. However, using an open corpus which has not been created specifically for the recommender system, requires selecting the most appropriate features for the recommendation scenario from the available corpus, which characterises the open corpus problem [17]. Besides selecting appropriate features in an open corpus of data and knowledge, solving it requires making metadata and user models re-usable, portable and interoperable between recommender systems [17].

To illustrate the problem of selecting features in an open corpus, consider the subset of musical artists on dbpedia. In August 2009 there were 23000 musical artists on dbpedia. In order to select a musical artist by its meta-data for a content-based recommendations, an interesting property might be the skos:subject of the artist. This reflects the categories of the artist on wikipedia. However, using the skos:subjects is not possible without selecting the most relevant categories, as there are a total of 15500 categories for the the 23000 musical artists. This contains a large number of categories which apply only to one artist, and which will not be useful for the content-based recommendation algorithm.

We illustrate the open corpus problem with an example in section 5, and then describe approaches for addressing the open corpus problem by using feature selection in the context of data mining algorithms and soft case-based reasoning. An additional use case which addresses the open corpus problem is describe in section 6.3.

3.4 The scalability problem

The recommendation algorithms used by closed corpus recommender systems are only scalable in a limited way [59], which makes them unsuitable to an open corpus with potentially millions of items or users. Current research on making the algorithms and data structures used for recommender systems more scalable, reflects the increasing importance of scalability for providing personalisation.

Early research with evaluations of collaborative filtering algorithms used 35,000 users and 3,000 items in the case of MovieLens [60], and 4,000 users and 1,600 items in the case of EachMovie [14]. Roughly ten years later in 2009, the largest research data set from the MovieLens project has grown to 10,681 movies and 71,567 users. Contrast this to commercial data sets from Amazon.com, which was already using data from 29 million users and several million items in 2003 for a collaborative filtering algorithm [43].

Addressing the scalability problem requires using algorithms of low computational cost, which make use of de-coupling the pre-processing phase from the on-line recommendation phase. Ideally, incorporating new data should not require re-computing of all available data. In addition, implementing a scalable open corpus recommender system requires a scalable architecture, which allows de-coupling providers of recommendation data from computational services and end-user facing services.

The architectural and technical principles of the Web of Data which are required to address the scalability problem are discussed in subsection 4.2. We then analyse the architectural requirements of scalable and open adaptive personalisation on the Web of Data, by presenting a reference architecture for Semantic Web applications in section 8.

4 The Web of Data as an open corpus

As we have outlined in the previous section, recommender systems can benefit from using an open corpus to provide the data for adaptive personalisation. We propose to use the Web of data as an open corpus. It contains sources from many different domains as well as cross-domain sources. As the Web of Data is based on Semantic Web technologies, these sources provide structured and semantically annotated data as well as ontologies providing background knowledge. This allows us to provide data and knowledge for the recommendation algorithms, thus mitigating the data and knowledge acquisition problems.

In this section we first describe the evolution of the World Wide Web towards Web 2.0 and the Web of Data and Web 3.0. Then we outline the principles behind the Web of Data, especially the Linked Data principles, and we give an overview of the relevant Semantic Web technologies. Finally, we describe the different available data sources and their domains.

4.1 The evolution of the Web

The World Wide Web established itself as a transforming force for all areas of society during the late 1990s. The foundation technologies of the Web allowed media, publishers, government, businesses and private individuals to publish any type of information to the Web, thus making it available instantaneously to anybody with a web browser. This period of the Web was characterised by static content created by large organisations such as government web sites, colourful brochure sites for multinational companies and e-commerce sites like ebay. This period is sometimes retroactively referred to as Web 1.0.

In the early 2000s however, many of those characteristics became transformed. A new generation of web sites was emphasising dynamic content and user participation. Users were encouraged to generate content such as blog posts and wiki entries, upload their own pictures and videos, and annotate existing web pages with their own tags and comments. Social networking sites enabled new ways for people to interact and stay in touch. The term Web 2.0 was coined to describe this new generation of web sites [51]. Examples of the most influential web 2.0 sites include Flickr for picture sharing, Delicious for tagging and bookmarking of web pages, YouTube for user generated videos and Wikipedia for a user generated encyclopaedia. All of these sites include elements of social networking sites as well as the means for creating user generated content.

The paradigm shift to Web 2.0 during the early 2000s was induced by the uncontrolled uptake of new principles and goals by the Web community at large. At the same time the World Wide Web consortium was developing a new set of technologies for the Web together with a scientific community. The goal of these new technologies was to simplify knowledge-intensive applications, by enabling a Web of interoperable and machine-readable data [8] based on formal and explicit descriptions of the structure and semantics of the data [24]. The umbrella term for this new set of technologies is Semantic Web. The benefits of Semantic Web technologies include simplification of information retrieval [1], information extraction [23] and data integration [26].

Due to the inherent complexity of implementing applications based on the Semantic Web technology stack [37], the uptake of the Semantic Web was limited from 2000 to 2005. Only after specifying the Linked Data principles as a lowest common denominator, a significant uptake of standards such as RDF started. The Linked Data principles have been adopted by an increasing number of data providers, especially from the Linking Open Data community project[4] which makes free and public data available as linked data. Figure 3 shows the growth of the number of Linking Open Data sources from July 2007 to July 2009. At the time

http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData/
Figure 3: Growth of the number of Linking Open Data sources: (a) July 2007, (b) April 2008, (c) September 2008, (d) July 2009
of writing, more then 90 different sources following the Linked Data principles are available. We discuss the Linked Data principles and Semantic Web technologies in more detail in the section 4.2.

The development of a global information space consisting not just of linked documents but also of linked data, has resulted in the emergence of the Web of Data [10] as a subset of the World Wide Web. Taken together, all linked data constitutes the Web of Data. While the World Wide Web provides the means for creating a web of human readable documents, the Web of Data aims to create a web of structured, machine-readable data. Semantic Web technologies provide the infrastructure for the Web of Data.

Some experts believe that the emerging of the Web of Data will provide the foundation for new business models and user communities, which together will constitute the next generation of the Web in the decade starting in 2010. This next generation is already sometimes referred to as Web 3.0, although only the foundations of this future Web currently exist.

4.2 Principles of the Web of Data

The Web of Data is a new initiative, which aims to lower the entry barriers for using and implementing Semantic Web technologies. As such it is focused on a subset of existing Semantic Web standards. However it describes how to integrate this subset with existing standards from the World Wide Web, in order to enable the Web of Data to be a true subset of the Web. This integration is described by the Linked Data principles.

The Web of Data utilises a subset of technologies from the Semantic Web technology stack: the Resource Description Framework (RDF) [24] provides a graph based data model and the basic, domain independent formal semantics for the data model. In RDF, data is expressed as subject-predicate-object triples and sets of triples form graphs, thus leading to the graph based data model of RDF. The SPARQL Query Language allows querying RDF data with graph patterns, and provides basic means for transforming RDF data between different schemata. In addition, technologies from the World Wide Web provide the fundamental infrastructure: Uniform Resource Identifiers (URIs) are used to provide globally unique identifiers for the data, and the HyperText Transfer Protocol (HTTP) is used for accessing and transporting the data.

In order to build a single Web of Data, all data providers have to follow the same guidelines for publishing their data and connecting it to other data sources. These guidelines are provided by the Linked Data principles [10], which specify how to use the different standards of the Web of Data together:

1. Use URIs as names for things (and e.g. persons, places).
2. Use HTTP URIs so that people can look up and access those names via HTTP.
3. When someone looks up a URI, provide useful information, using the standards RDF and SPARQL.
4. Include links to other URIs, so that data about more things can be discovered.

In addition to the LD principles, guidelines and best practices for publishing, discovering and consuming Linked Data are currently developed. [11] describes guidelines for publishing different types of RDF data in accordance with the Linked Data principles, and [61] describes criteria for consistent and persistent naming of URIs. We describe the architecture of Semantic Web applications, and the most common functionality for discovering and consuming RDF data in section 8. An overview of other emerging principles is given in [10].
4.3 Available Linked Data sources

As Figure 3 shows, currently there are already linked data sources from many different domains available. They provide data which ranges from place names (there are at least 28 cities called “Paris”), to citation counts for scientific publications, user generated review scores for restaurants, protein names and all BBC broadcasts of the “SciFi” genre. Going beyond the LOD cloud, public government data will be offered, and it is expected that many e-commerce sites will provide Linked Data in the near future, as major search engines have started indexing semantically annotated web pages.

The nucleus of the linked data cloud is formed by DBpedia, which extracts RDF from wikipedia topic pages, and thus provides URIs and RDF data about topics from any domain. DBpedia reuses properties from existing vocabularies such as SKOS (Simple Knowledge Organisation System), but it also introduces its own terms for e.g. expressing that a Banana Split has the flavour of a Banana with \texttt{dbprop:flavour}. By providing URIs and semantically structured information about any kind of subject, dbpedia is providing authoritative URIs. This allows other services to reuse these URIs and describe their own resources in terms of their relationship to resources from dbpedia. For example a blog post may have the topic of http://dbpedia.org/resource/Fencing.

Social Web sites provide a big contribution to the linked data cloud, by making user generated content from their users available. This data is modelled after the principle of object centred sociality [13]. Such data uses the Friend of a Friend (FOAF) vocabulary for describing users and their connections to interests and other users, and the Semantically-Interlinked Online Communities (SIOC) vocabulary for describing user generated content on forums, weblogs and web 2.0 sites, as described in [13]. This kind of data is provided e.g. by LiveJournal. Using FOAF and SIOC data already enables the vision of portable user identities and reuse of a user’s social graph to a certain extent [12].

Broadcasters and news publishers provide data about media content and usage. As an example, the DBTune project [56] makes such data about users and their connections to musical artists available. For every user, the top artists to which the user listens are available as linked data. At the time of writing, the DBTune Myspace data contained at least 6 million user-item connections. This data is well suited for statistical algorithms and collaborative filtering, however it does not provide structured knowledge about the artists (e.g. the genre). However such knowledge can be obtained by linking the artist to another LD source, such as DBpedia or Musicbrainz. Other sources include the BBC’s catalogue of broadcasts on TV and Radio.

Place names and geographic co-ordinates are made available by Geonames and census data is available in parts from the US and the EU. Data about scientific publications is made available by many providers, including the IEEE, ACM and CiteSeer. The LD principles have been very popular in the domain of life sciences, due to the knowledge intensive nature of much research in this domain. Some of these sources include UniProt, PubMed and the Gene Ontology.

E-commerce sites can use the Good Relations vocabulary[3] to describe their products and their features and prices, payment options, as well as store locations and opening hours. BestBuy has released such data about all of their stores in the US.

Many governments are increasingly making public government data available in a digital form. This includes statistics about topics such as agriculture and life expectancy. While the UK government makes
such data directly available as linked data, the US government provides their data in different legacy formats such as spreadsheets and text files. Parts of the US data are then converted to linked data by the community.

It is very likely, that both the number of sources and the number of domains are going to grow as two major search engines have announced their support for RDF data. Yahoo has introduced RDF support in its SearchMonkey search engine infrastructure. Google will show RDF information along with traditional search results as “rich snippets.” The inclusion of RDF data in search engine results is expected to make RDF a requirement for search engine optimisation. In order to get a high ranking for a specific search term, a web site will have to provide semantic annotations for its content. This incentive will lead to a further growth of the Web of Data.

5 Selecting features of Linked Data

The Web of Data is characterised by the links between the different sources. While relational databases rarely contain links to external data, RDF data can easily provide relations between identifiers from different data sources. In addition to links between different sources, the Web of Data provides connections between related content from different domains and links between identical users in different communities. Such data has not been available for the purpose of making recommendations before, and as such provides both new opportunities as well as new challenges.

While the previous section has shown that the web of data provides data and knowledge which can be used as an open corpus, in this section we are going to explain why the intrinsic links in Linked Data allow us to address the open corpus problem. These links between different sources allow us to go beyond the capabilities of current recommender systems. By using connections between resources from different sources, domains and communities, we can provide cross-domain recommendations. However, before such data can be used for a recommendation algorithm, appropriate data sources and appropriate data subsets need to be selected. The pre-processing of data to identify the subsets which are relevant for the recommendation scenario is called feature selection.

In this section we first explain the nature of the intrinsic links on the Web of Data through an example. Then we describe the task of feature selection and the most relevant techniques for feature selection on linked data in the context of adaptive personalisation. This includes the techniques of dimensionality reduction and spreading activation. We conclude the chapter by explaining how soft case-based reasoning (CBR) can be applied to reasoning on features of large scale structured data such as the Web of Data.

5.1 Following links between data sources

Figure 4 shows an example of following the links between sources in order to provide recommendations. In this example we are going to use two sources: The social network MySpace which has a special focus on facilitating connections between musicians and their fans through social networking tools. The DBTune

http://data.gov.uk/
http://www.data.gov/
http://data-gov.tw.rpi.edu/
http://developer.yahoo.com/search/boss/structureddata.html
http://knol.google.com/k/google-rich-snippets-tips-and-tricks
http://www.myspace.com/
Figure 4: Following links between sources to provide recommendations

The project[^10] provides a live representation of MySpace users and their connections to musicians as RDF Linked Data. The other data source is DBpedia, which provides Linked Data for the info-boxes from Wikipedia pages. As such, DBpedia provides data about all the different domains which can be found on Wikipedia.

Using DBTune as a source on its own, lets us only get data about MySpace users and their musical preferences. The musical preferences are expressed in the links from a user profile to the top artists the user is listening to. This allows us to use collaborative filtering to match users with similar preferences to each other. For instance, “Kyle Butler” listens to Johnny Cash, Elvis and John Cage, while “TheTeacher” listens to Johnny Cash, Elvis and Metallica. There is an overlap between the musical preferences of both users as far as American popular classics are concerned.

The Web of Data allows us to discover links between user identities, which can indicate that two users in different communities are identical. For instance, such links might be contained in a FOAF profile as RDF, or they could be directly specified as part of either the MySpace or Wikipedia user profile. Having such a link between **identical users in different communities**, allows us to infer that “KyleButler” on MySpace and “Dexter Morgan” on Wikipedia represent the same user.

We can now use this link between the two identical users to follow a **link between different data sources**. By using the link from “KyleButler” to “Dexter Morgan”, we can follow it from MySpace to Wikipedia, thus providing us not just with the musical preferences of this user but also with the Wikipedia topics, which the user has edited. From editing these topics, we can infer a certain degree of expertise in these topics. “Dexter Morgan” has edited the country page for the Netherlands, the page for the city of

[^10]: http://dbtune.org/myspace/
Amsterdam and the page for the sailing sport.

Now as a third step, we can use all of these explicit links to infer implicit connections between different domains. “KyleButler” has shared musical preferences with “TheTeacher”. By following the link between these user identities, we could recommend “Dexter Morgans” topics to “KyleButler” on MySpace. The specific recommendations would depend on the recommendation algorithm and on the recommendation domain. E.g. for a travel scenario we could create a holiday itinerary of travel destinations consisting of country and city pages which were edited by wikipedia users which share a MySpace users musical preferences. This itinerary then could include the Netherlands and Amsterdam. Another possible scenario would recommending books about sailing to “KyleButler” if a statistical significant connection between people listening to Elvis and Johnny Cash and people who have an interest in sailing can be found.

5.2 Feature selection

If we transfer the example from figure 4 to real data from DBtune and DBpedia, then the number of links and topics which provide potential candidates for a recommendation surpasses the human potential to provide a hypothesis as to why something should be recommended. As such, this represents a data set which is well suited to feature selection as it exhibits the following problems: there are to many features, and individual features are unable to independently capture significant characteristics of data. On the other hand there might be a high dependency between some features, and there could be emergent behaviours of combined features.

Feature selection can reduce the dimensionality of a data set in order to enable data mining algorithms to work effectively on data with large dimensionality. It is a process that selects a subset of original features, based on an evaluation criterion. This process usually has four steps: (1) generation of a subset of all potential features; (2) evaluation of the subset based on some sort of criterion; (3) application of a stopping criterion, as soon as the performance of the selected feature is sufficient; (4) validation of performance of the selected feature subset on live data. The evaluation criteria are based on measuring the effect of adding or removing features from the subset.

There are three main algorithmic schemes for executing the four steps of a feature selection process: filter algorithms, wrapper algorithms and hybrid algorithms. Filter algorithms start with a subset of the whole data set and then search through the feature set by a particular search strategy. Each generated subset is evaluated by a measure, and the evaluation result is compared to the previously best result. The search iterates until a predefined stopping criterion is reached. Wrapper algorithms are very similar to filter algorithms. However, instead of a measuring function, a predefined mining algorithm is used to evaluate each subset. This has the benefit of choosing the feature set which is best for a specific data mining algorithm, however the drawback is the higher computation cost compared to generic measures used in filtering algorithms. Finally, hybrid algorithms combine aspects of filter and wrapper algorithms to lower the total computational cost. Generic measures are used to find good candidate subsets. Then these subsets are further evaluated with a specific data mining algorithm.

These feature selection algorithms can be used for pre-processing data for many different data mining algorithms, especially for classification and clustering. Another data mining algorithm, which can benefit from feature selection is Latent Semantic Indexing (LSI). LSI is especially well suited for data sets with a high dimensionality of features, such as the RDF properties of Linked Data sources. However, LSI is characterised by a high computation cost. As suggests, by using a state vector machine to pre-select features before applying LSI, it is possible to significantly reduce the computational cost of dimensionality reduction.
An alternative approach to the four step feature selection process, is to use spreading activation for pre-selecting a set of features. It has been used in semantic graph representations of social networks for the purpose of recommendation\cite{44}, as well as for ranking and navigating \cite{41}. Spreading activation is well suited for graphs with a high connection density, in which most elements are connected to most other elements. In order to implement spreading activation, the flow of light through the connections is simulated. After activating a set of start nodes, light travels from node to node through the connections. However each node lowers the intensity of the spreading activation, until it has crossed a threshold and is unable to further activate a node. In this way the activation spreads through the network. The resulting nodes with the highest activation level are then selected from the whole graph.

5.3 Case-based reasoning

Case-Based Reasoning is a complimentary approach for exploiting the intrinsic links between data sources on the Web of Data. While feature selection allows for pre-processing as part of data mining, CBR is a completely orthogonal methodology of using structured data for reasoning. CBR is a methodology for solving new problems based on the solutions of similar problems in the past \cite{69}. For using Linked Data, this translates to viewing all data about one resource from a users profile as a case. We then can use the cases which are associated with a user to find similar but new resources, which can be presented as recommendations to the user. In this way, CBR provides a methodology for reasoning on large scale structured data, such as the Web of Data.

Case representations can include three parts: the problem, the solution for the problem and the outcome after applying the solution. Case-based reasoning has been formalised as a four-step process \cite{69}:

1. **Retrieve**: given a target problem, retrieve relevant cases for solving it, taking the outcome of the solutions into account. In order to retrieve relevant cases, data mining techniques such as clustering and models with latent variables can be used.

2. **Reuse** the cases to attempt to solve the problem, this might involve adapting solutions to fit a new problem.

3. **Revise**: after observing the outcome of the reused solutions, revise the solution if necessary.

4. **Retain**: After revising the solution successfully to the target problem, store and save the resulting case for future reuse.

Case-Based reasoning is an alternative to other reasoning strategies \cite{58}. Rule based reasoning is based on formal representation of rules, and is very inflexible if a problem description does not match a rule. In comparison, case-based reasoning is a form of reasoning by analogy. By identifying commonalities between retrieved cases and the target problem, CBR forms a generalisation of the solution from multiple successful solutions. Therefore CBR tends to be a good approach for complex domains, in which case descriptions tend to be very unique for each case.

**Soft Case-Based Reasoning** takes the implicit comparison of cases and their solutions one step further, by introducing soft computing techniques to CBR \cite{67}. Soft computing techniques have been developed to have a tolerance for imprecision, uncertainty, approximate reasoning and partial truth. This allows them to contribute to the different steps of the CBR process in order to make it more robust. **Fuzzy logic** allows approximative comparison of sets and numerical features, which can be used for case matching and retrieval in CBR. **Artificial Neural Networks (ANNs)** are used for learning and the generalisation of knowledge and
patterns. This can be used to help retrieve relevant cases when only a subset of all features are relevant for one specific new target problem. Genetic algorithms are used to solve search and optimisation problems, inspired by the biological principles of natural selection and genetics. They can be used in CBR to learn local and global weights of case features. These weights indicate how important the features within a case are with respect to the solution features.

Distributed case-based reasoning The vast majority of CBR systems have taken a single-agent, single-case-base approach to problem solving [55]. However, while this approach has been proven to be effective in certain application settings, there are two important factors that may render this approach unsuitable: (1) privacy and (2) scalability [55]. Privacy refers to the situation when cases are stored at a system, and if the cases can not be shared with other systems or stored at a central case base. Scalability refers to the need of distributing the cases across systems so that large amounts of data can be better processed. Approaches to make case-based reasoning distributed involve the combinations of a single agent operating on multiple case bases, multiple agents operating on a single case base, or multiple agents operating on multiple case bases [55].

In order to use distributed case bases which use heterogeneous schema to describe the knowledge contained in individual case containers, it is necessary to create a shared index which allows aligning the vocabulary of different sources [5]. Approaches from peer to peer research about query routing and data peer management systems provide possible candidates for this [32].

6 Use cases

In this section we introduce use cases to illustrate the benefits of providing adaptive personalisation by using the Web of Data as an open corpus. Data which connects users from different communities and topics from different domains has never before been available on web scale. This opens up new use cases for applying the discussed approaches for personalisation and recommendation.

While the previous sections have focused on providing the background on adaptive personalisation and recommendation, this section will provide a narrative to explain the end-user benefits which arise from exploiting linked data for personalisation.

First we describe how the Web of Data can be used to mitigate the data acquisition problem in order to recommend music based on collaborative filtering. The implementation and evaluation of this use case is described in section 7. Then we outline how soft case-based reasoning can be used to to mitigate the knowledge acquisition and the open corpus problem for expert finding on the Web of Data. Finally, we sketch how feature selection and data mining could enable a context based browsing agent, which recommends trails of web sites which are relevant to the current context of the user. The requirements of this use case include feature selection to mitigate the open corpus problem and a scalable architecture.

6.1 Using collaborative filtering to recommend music

Providing personalised on-line music streaming services is one of the most popular applications of recommendation algorithms. While the first such service was the Smart Radio academic prototype [34], these services have reached a stage of maturity, which allows them to charge for their service. The most popular examples of such services include Last.FM [1], Pandora [2] and GrooveShark [3]. These are established services

http://last.fm
http://pandora.com
http://grooveshark.com
in this domain, which makes it necessary for new service providers to offer recommendations of at least the same quality as the existing services.

In order to acquire sufficient data to provide recommendations with a competitive quality, a new recommendation service can use the Web of Data which contains publicly accessible data about musical preferences. This data is available from social networking sites, which use the principle of object centred sociality to model the indirect connections between users via their interests. One source which provides such data is MySpace via the DBtune project, which contains data about the top artists for each MySpace user. In addition to this social data, we can use DBpedia to provide genre information and to provide connections between different artists, e.g. which artists collaborated in a band at one time.

By using the intrinsic links contained in the linked data from DBtunes and DBpedia a new music streaming service can mitigate the data acquisition problem: by linking the musical artists in the inventory of the new service to DBpedia, and then following the links from DBpedia to MySpace, all users which are connected to at least one musician from the services’ inventory can be harvested. This allows building a densely populated matrix of background data for a collaborative filtering algorithm.

We describe the details of using such data for collaborative filtering, evaluate the results and discuss related work in Section 7.

6.2 Using case-based reasoning for expert finding

Social networking and user generated content are not only used in consumers leisure time, they can also be used for connecting employees in knowledge centred enterprises with each other. This use of Web 2.0 paradigms is called Enterprise 2.0 [47]. While social networking sites such as Facebook are used to connect with old friends or possible dates, business oriented social networks such as LinkedIn [4] and Xing [5] allow searching for people based on their expertise and work experience.

The Web of Data also shows potential to be used for the task of expert finding. Social networking sites such as LiveJournal [6] provide RDF versions of user profiles. The profiles are exported using the FOAF vocabulary, and include a wide variety of interests, from sports over youth cultures to local politics. In addition, they can also include work related qualifications such as programming languages and W3C standards, which makes them interesting for expert finding.

Although RDF prescribes the data model, and FOAF prescribes the vocabulary used for user FOAF profiles, different profiles from different social networking sites use the vocabulary slightly different. In order to use these profiles for recommending exports for certain topics, we could use soft case-based reasoning, which would be unaffected by the slight variations in the usage of the FOAF vocabulary. If each FOAF profile represents a case, then we can use fuzzy matching techniques for finding profiles related to a specific topic of expertise. Soft case-based reasoning and fuzzy matching are discussed in section 5.3.

In addition, the case-based reasoning system for expert finding from distributed sources can be implemented as a distributed case-based reasoning system [55]. Instead of integrating the FOAF profiles before using them for the CBR process, each case base can define mappings to a few other case bases. These mappings then could be used for distributing queries for specific expertise to the right case bases [32].

This use cases illustrates, that by using the Web of Data to provide personal profiles which include topics of expertise, we can mitigate the knowledge acquisition problem in order to provide recommendations.

[^4]: http://linkedin.com
[^5]: http://xing.com
[^6]: http://livejournal.com
Based on expert finding. In order to select the features of the personal profiles which are most relevant to the task of expert finding, we can employ soft case-based reasoning, thus \textit{mitigating the open corpus problem}.

6.3 Providing a context based browsing agent

The first visionary ideas for using information technology to handle large amounts of documents, data and knowledge, such as Vannevar Bush and his Memex idea, already contained the idea of recording trails through information. Current social bookmarking sites such as Delicious allow finding pages related to a bookmark via its tags. However, they do not allow saving sequences of bookmarks.

New developments, such as the TrailMeme\footnote{http://trailmeme.com/} site, allow the arrangement of bookmarks to form trails which can include branches. This allows recording a list of bookmarks which provide an overview of a specific topic, where the user can follow a trail of sequentially dependent topics with the option to choose between different alternatives during the progression of the trail. This allows e.g. providing a list with introductory links to the Java programming language, with the possibility to decide which specialised area of Java is most interesting to the user: e.g. network programming, graphic programming or security programming.

While this goes towards allowing users to save and share trails of bookmarks, it does not provide any support based on the current context of a user. If a user is interested in finding a trail of sequentially related information for an urgent and unexpected information need, then he still has to use a search engine. In addition he will have to discover the inherent dependencies between the topics on his own.

As an example, consider a knowledge worker, Mr. K., who has to change some last minute details in the user interface of a prototype immediately before the presentation of the prototype. The person who implemented the prototype is currently in transit for the presentation or he is unavailable for a different reason. Mr. K. only needs to exchange the logo of one of the partners. Mr. K. is a very experience programmer, however he has no experience with the programming language used for the user interface (some obscure scripting language), nor the library used.

If the original programmer would have been able to record the web sites and user manuals which he was reading in order to implement the user interface, then a subset of this whole set of trails could be recommended to Mr. K. in order to solve his urgent task. Instead Mr. K. will have to relay on his experience, on manuals and on a search engine.

In order to support an organisation in situations like this, a context based browsing agent could be employed. This agent could record the work related browsing trails of all the users in the organisation on a voluntary basis. Then this corpus of trails could be used to recommend sites, pages and trails to users, which try to solve problems which already have been solved by their co-workers. Especially in large business organisations or large communities of people with the same interests this could be a very effective means of information filtering.

In order to implement such a context based browsing agent, the challenges of the \textit{open corpus problem} and of \textit{scalability} are most important. After collecting the web sites which where part of the trails, which content features are suitable for describing the context of the user at the time of pursuing his trail? In addition the architecture of the background services needs to be scalable, as the number of users contributing their trails can rise quickly and the personalisation algorithms need to scale with the number of users. A suitable approach might be employing filter algorithm based feature selection in conjunction with latent semantic indexing, due to the high dimensionality of the collected information.
7 Applying collaborative filtering to the Web of Data

In this section, we describe how the Web of Data can be used to mitigate the data acquisition problem and thus allow a new recommendation service provider to compete with existing services. We describe how a music recommendation service can augment its data by linking musical artists to DBpedia and then to MySpace data, and then harvesting data about users musical preferences. This shows that Linked Data can enable providing recommendations for new users, it can improve recommendations by reducing the sparsity of the background data, and it can be used to add new items to the recommender system.

We first describe the details of how to integrate Linked Data for collaborative filtering algorithms using SPARQL, then we describe how this allows us to mitigate the different parts of the data acquisition problem: new users, new items and the sparsity of background data. We then describe the results of a preliminary evaluation, which augments real data from a small scale, closed streaming radio service with Linked Data. This significantly improves the precision and recall of the recommendations when compared to Last.fm recommendations for the same artists. Finally, we conclude the section by presenting related work.

The presented prototype is based on the use case described in section 6.1.

7.1 Integrating linked data for collaborative recommendations

As explained in section 2, recommender systems usually consist of three components. In order to use Linked Data, we need to extend the architecture of the recommender system with two components: the data interface allows accessing URIs via HTTP in order to acquire RDF data. This provides an abstraction layer for accessing the data. RDF libraries such as Redland provide a data interface component. The integration service transforms the data from the different sources into a unified representation, which we describe in the next section. For a detailed discussion of the components of Semantic Web application see section 8.

Figure 5: Processing Linked Data for Collaborative Recommendations

*This section is based on Heitmann and Hayes 2009 [35]

[http://librdf.org/]
Figure 5 shows all the steps of processing Linked Data for collaborative recommendations: (1) **integrating** the data about user-item connections from different sources to a common vocabulary. (2) **Transforming** the representation of the data from an RDF graph to a user-item matrix. (3) **Applying** a specific collaborative filtering algorithm on the user-item matrix.

**Integrating** data is necessary, because different data sources will use different vocabulary to denote the connection between a user and an item. However, as we are only interested in those connections, extending the integration service only requires writing one new rule. Figure 5 shows the vocabulary used by two example data sources: One source uses the `foaf:Person` class for the users, `mo:MusicalArtist` for musicians and `myspace:topFriend` to connect both. The second source uses a different vocabulary to express the user-item connections, `foaf:Document` indicates the page with user information, `sioc:links_to` describes a link from that user page, and `sioc:WikiArticle` is a wikipedia article about the topic the user is interested in. By using SPARQL CONSTRUCT queries, we can instruct the data interface to fetch the data and then provide a rule for expressing the data with a different vocabulary.

This is the query for transforming the first data source to FOAF:

```sparql
CONSTRUCT { ?user foaf:interest ?item }
WHERE { ?user myspace:topFriend ?item .
  ?user rdf:type foaf:Person .
  ?item rdf:type mo:MusicalArtist }
```

The resulting RDF graph of user-item relations then needs to be **transformed** into a user-item matrix. Columns represent users, rows represent items, and their connection is indicated by a binary value. Representing the user-item connections as a matrix is necessary, because the user-item matrix is typically required by collaborative filtering algorithms for calculating similarity between users or items.

Finally the collaborative filtering algorithm can be **applied** to process the user-item matrix into its background data, which might be an item-item matrix, a user-user matrix or any other data structure.

### 7.2 Mitigating the cost of data acquisition

Collaborative filtering algorithms require sufficient data about users and items in order to make recommendations. We can mitigate the data acquisition problem for collaborative filtering algorithms by utilising Linked Data sources. They provide data which can be used to “fill in the gaps” in the background data.

**Providing recommendations for new users:** Consider the use case of a wikipedia editor creating a new account on a music recommendation web site. If a new user joins a web site in order to receive recommendations, then there is no background data available about the preferences of the user. However, if we can find Linked Data about the user, which connections him to the musical artists in the background data of the recommender system, then we can instantly provide recommendations for him. By using the SIOC MediaWiki exporter we can access SIOC data for a wikipedia editors homepage. This will include links to different topics for which the user indicates interest or to which the user has contributed. If the recommender system has background data about one or more of these topics, then it can add one row to the user-item matrix, and provide recommendations for the user.

**Adding new items to the inventory:** A recommender system can not recommend a new item until it has been rated by other users. Using e.g. Linked Data from DBTune about MySpace musicians, we can add data about the new musician, without waiting for users to add the item to their preferences. One
way to do this, is to collect all users from MySpace which are connected to the new musician and to at least one musician for which the recommender system already has background data. If we add this data to the user-item matrix, there will be one new row for the item and multiple new user columns. The anonymised data of these users from MySpace allows us to add indirect connections between the new item and existing items.

**Improving recommendations by reducing sparsity**: If the number of connections between users and items is low, compared to the total number of items, then the number and quality of recommendations will be low. In order to discover similarities between items, the user-item matrix should provide as many indirect connections between items as possible. One way to do this, is by adding new users from a relevant external source, e.g. from the DBTune MySpace data which are connected to more than one musician. This will add multiple user columns, which add to the total number of connections between users and items. This in turn provides more background data for the collaborative filtering algorithm to discover similarities.

### 7.3 Preliminary Evaluation

To demonstrate the validity of our approach, we augment the data from a real collaborative music recommender system with Linked Data, and significantly improve its precision and recall.

Smart Radio was the first on-line music streaming and recommendation service [34] which operated on similar principles to Last.fm but pre-dated it. The Smart Radio used a closed database as background data for the recommendation algorithm. However as a research proof-of-concept it only had a limited number of users and items: 190 users and 330 musical artists.

We have expressed the user-item connections of the Smart Radio background data as Linked Data using the FOAF vocabulary. The users have been anonymised for this, and all musical artists have been linked to both DBTune MySpace data and DBpedia. We then used the links from Smart Radio artists to MySpace artists to reduce the sparsity of background data, as explained in the previous section. This added 11000 users from MySpace and around 25000 new connections between users and artists. We then created user-item matrices for the Smart Radio FOAF data and the combined FOAF data from Smart Radio and MySpace.

As recommendation algorithm, we applied a binary cosine similarity measure from [59]:

$$\text{cosine}(i_1, i_2) = \frac{\text{count}(i_1, i_2)}{\text{count}(i_1) \text{count}(i_2)}$$  \hspace{1cm} (1)

This is a very simple baseline recommendation algorithm, which can be used to compute an item-item similarity matrix. Each entry in the item-item matrix expresses the similarity between two items. \(\text{count}(i)\) is the number of users who have a connection to item \(i\), and \(\text{count}(i_1, i_2)\) is the number of users who have a connection to both item \(i_1\) and item \(i_2\). Ranking the entries in row \(i\) provides the most similar items to \(i\).

In order to evaluate the recommendation results, we compared them to the last.fm recommendations for the same artist: \(B_{\text{artists}}\) is the set of all artists in the background data, \(R_{\text{lastfm}}(a)\) is the set of last.fm recommendations for artist \(a\). Then \(D(a) = B_{\text{artists}} \cap R_{\text{lastfm}}(a)\) is the set of relevant recommendations for artist \(a\). For a second recommendation \(R(a)\) we can then define precision and recall [38]: Precision is the number of relevant artists in a specific recommendation divided by the total number of recommendations, while recall is the number of relevant artists in a recommendation divided by the number of relevant artists in our background data:

$$\text{precision}(R(a)) = \frac{|R(a) \cap D(a)|}{|R(a)|}$$  \hspace{1cm} (2)
\[ \text{recall}(R(a)) = \frac{|R(a) \cap D(a)|}{|D(a)|} \] (3)

**Results:** Computing the average recall and precision for both data sets, shows that augmenting the Smart Radio data significantly improves the recommendations. Using only Smart Radio data we get an average precision of 2\% and average recall of 7\%. By augmenting the Smart Radio data with Linked Data from MySpace we get an average precision of 14\% and average recall of 33\%. This increase in the relevance of the recommendations shows that our approach is a viable first step.

### 7.4 Related work

“Foafing the Music” [21], might be the first recommender system for music which used RDF data. In order to provide its recommendations it crawled data from a large number of web sites, such as Amazon, Eventful.com, music newspapers, and music blogs (about 1100 different sources). The user preference data is stored as FOAF data. The recommendation algorithm combines knowledge-based and content-based approaches. No evaluation of the recommendation results is performed. While this recommender system saves preferences as FOAF data, it does not use RDF data or Linked Data as background or input data of the recommendation algorithm.

[54] describe the Linked Data from the music domain, which could be used for recommendations. However the suitability of Linked Data for different recommendation algorithms is not discussed, and no implementation is described. [57] describe their work on the Music Ontology, which can be used to encode meta-data about music and connect it to the LOD cloud. Based on this, an approach for providing content-based recommendations is discussed, which uses not only meta-data but also the audio signal of the music for recommendations.

[65] compares the efficiency of recommendation systems built using data from the public web as background data, with recommender systems using closed, private data sets. Search engine and web crawler results are used to build a user-item matrix for a collaborative filtering algorithm. The recommendation results are evaluated against the results of recommender systems from the same domain with a private, closed data set. The evaluation shows that the results are on an equal level. However, the collaborative filtering algorithm does not use RDF or Linked Data as background data. [7] introduces different approaches for exchanging data about users or items between collaborative recommender systems. However, the underlying technology for storing and exchanging data is not discussed.

### 8 An architecture for open and scalable recommender systems on the Web of Data*

Using the Web of Data as an open corpus for personalisation and recommendation, introduces new challenges for the scalability of the recommender system. Adding multiple external data sources to a recommender system, adds new requirements for the system such as aligning the schema of external sources and integrating the available data before it can be used. This new functionality must be scalable enough to handle the amount of data which is available from the external sources.

Scalability is a cross-cutting concern, which affects all parts of a system. Such cross-cutting concerns are addressed by changing the software architecture of a system [2]. Every software architecture is an

---

*This section is based on Heitmann, Hayes and Oren 2009 [65] and on Heitmann, Kinsella, Hayes and Decker 2009 [37]
abstraction of a runtime system, which specifies the roles of the different parts of the system and how they interact [27]. In order to implement the functionality which is required for using the Web of Data as an open corpus in a scalable way, an open recommender system requires a different architecture than a closed recommender system.

**Figure 6:** Research approach for identifying the architectural requirements of an open recommender system which uses the Web of Data

Figure 6 outlines our research approach for specifying the requirements of a scalable and open architecture for recommender systems:

- the main **goal** is to enable the transition from *closed* recommender systems to *open* recommender systems which use the Web of Data as open corpus. In order to accomplish this, we need to specify an open and scalable architecture for recommender systems. We do this in three steps:

  - **First step:** compare closed recommender systems and closed web applications. This step is based on the literature survey in section [2]

  - **Second step:** identify the new **components**, which are required for open applications which use the Web of Data. These new components implement the functionality which allows the application to be
open to external data sources, and they constitute a software architecture. This step is based on an empirical analysis of 98 Semantic Web applications.

- **Third step:** specify how the architecture and components of open applications on the Web of Data can be applied to recommender systems on the Web of Data in order to provide a scalable and open architecture.

As we are going to show in this section, the architectural requirements of open corpus recommender systems mirror the requirements of Semantic Web applications:

**Open corpus recommenders** provide recommendations using public data sources which are relevant for the recommendation domain and scenario. Their architecture must take discovery, selection and integration of data sources without prior knowledge into account. The different steps of the recommendation process can be performed by different parties, in order to make the application scalable.

**Semantic Web applications** are decentralised and open, operate on distributed data that can be published anywhere, may conform to arbitrary vocabulary and follow semi-structured schemas [53]. In order to use external data sources with different schema, the application must discover, select sources and integrate the external data. The components of a Semantic Web application can be distributed across multiple hosts [52].

**Outline:** This section is structured as follows: In order to provide a formal foundation for the rest of the section, we provide our definitions of the terms “software architecture” and “Semantic Web application” in section 8.1 We also discuss the different abstraction levels of architecture, and existing approaches for the architecture of the Web and the Semantic Web.

After this, we summarise the similarities between closed recommender systems and closed web applications in section 8.2 (step 1), based on our literature review. This is followed by the presentation of our reference architecture (step 2) for applications using Semantic Web technologies in order to access the Web of Data, in section 8.3. Finally, we specify how the Semantic Web reference architecture can be applied to scalable and open recommender systems (step 3) in section 8.4. We explain how each component from the reference architecture can be used as part of an open recommender.

### 8.1 Defining software architecture for the Web of Data

Before we perform our architectural analysis of current Semantic Web applications, we will define the term “architecture”. Since the Semantic Web extends the World-Wide Web [9], we also review existing architectural approaches for the Web and the Semantic Web, and provide a definition for the term “Semantic Web application”

**Defining software architecture**

Software architecture provides an abstraction of a runtime system, which can be used to address cross-cutting concerns.

Software architectures are abstractions of runtime systems [27]: an architecture consists of components, connectors and data, and describes patterns for combining the components in a meaningful way; abstracting software systems into architectures is useful when discussing and analysing common behaviour across systems and when developing software frameworks that support application development by providing such common behaviour through off-the-shelf implementations.
Specifying a software architecture can be used to address **cross-cutting concerns** and design decisions which affect all parts of the system [2], and as such are very costly to reverse later on in the development of the system. Such cross-cutting concerns include security, maintainability, but also **scalability**. These concerns affect every part of a system, and can only be addressed in a very limited way by changing individual parts. These concerns can be best addressed on the architectural abstraction level.

**Different abstraction levels of architecture**

Using Fielding’s definition, every abstraction layer of a software system can have its own software architecture. Fielding explicitly differentiates between (a) architecture on the abstraction level of a single application, and (b) architecture on the higher abstraction level of network-based applications, “where the interactions among components are capable of being realised in network communication” [27].

Both the architecture of the World Wide Web and the different architectural approaches for the Semantic Web address the abstraction layer of network-based applications. While they have an impact on the implementation of individual applications, they do not provide an architecture for individual applications or any other support for software engineers wishing to leverage the benefits of either the World Wide Web or the Semantic Web.

**The architecture of the World Wide Web**

The document “Architecture of the World Wide Web, Volume One” [40] represents the result of an iterative process, in which best practices proposed by the community of practitioners and the standards proposed by the World Wide Web consortium informed each other over years. The document describes the overall architecture of the World Wide Web, how multiple standards have to be used together in this architecture, and which community guidelines exist for implementing different aspects of the architecture. A central part of the WWW architecture is Fielding’s Representational State Transfer (REST) architectural style [28], which prescribes a way for networked applications on the Web to interact.

**Architectural approaches for the Semantic Web**

In comparison, the evolution of best practices and standards for Semantic Web technologies has not culminated in a single architectural approach. As argued in [30], the Semantic Web “layer cake” of official standards can be seen as an architecture for the Semantic Web. A second architectural approach is represented by the Linked Data community best practices [11], which leverages the World Wide Web architecture and augments it towards supporting distributed publishing of RDF data. Other best practices e.g. for consistent and persistent naming of URIs [61] are emerging. The W3C task force “Architecture of the World Wide Semantic Web” [1] is concerned with providing an approach which encompasses all possible parts of an architecture for the Semantic Web.

**Defining Semantic Web applications**

The most basic requirement for a Semantic Web application is the use of RDF for the metadata used by the application. This can be derived from the fundamental role of RDF in the “layer cake” of Semantic Web standards [40]. Additionally a set of formal vocabularies should be used to capture the application domain, and **SPARQL** should be used as data query language, according to [33] (definition 2.2). All surveyed
applications meet these requirements, except for applications using programmatic access to RDF data for efficiency reasons.

8.2 Step 1: comparing closed recommenders and closed web applications

Our first step towards specifying an architecture for open and scalable recommender systems on the Web of Data, is to compare closed recommenders with closed web applications in order to identify similarities regarding their software architecture. Figure 6 contains a summary of these similarities.

Characteristics of closed recommender systems

Our literature survey on the topic of adaptive personalisation and recommender systems in section 2 has shown that recommender systems require three components to provide recommendations: (1) background data, which is the information the system has before the recommendation process begins, (2) input data, which is the information provided about the user in order to make a recommendation, and (3) the recommendation algorithm which operates on background and input data in order to provide recommendations for a user.

Closed recommender systems usually only use data collected in-house for the purpose of providing recommendations for a specific domain and scenario. This means that in most cases a recommender system is only using one data source so that no integration is required. If several in-house data sources are used, then all of these sources are under the control of the same organisation. This means that all schema of these sources are known in advance, so that integration is easy and cheap.

Characteristics of closed web applications:

As described in [42], the predominant pattern for closed web applications is the model/view/controller pattern, which results in a three tier web application architecture which implements the model through a relational database, the controller through business logic and the view through HTML.

As explained in [53], web applications modelled after this pattern usually only use one data source, which is their own relational database management system. The data for this database is created by the application itself, using one fixed schema, so that no integration is required.

Similarities between closed recommender systems and closed web applications

From an architectural perspective, closed recommender systems and closed web applications are very similar: the both only use one data source with one schema so that no integration is required. If multiple sources are used, then these are all from the same organisation so that all schema are known in advance, which makes the integration easy.

8.3 Step 2: Reference architecture for Semantic Web applications

Our second step is to identify the new components, which are required for open applications which use the Web of Data. These new components implement the functionality which allows the application to be open to external data sources, and they constitute a software architecture. This step is based on an empirical analysis of 98 Semantic Web applications. Figure 6 contains a summary of the differences between closed web applications and open applications using the Web of Data.
Software architecture can be used to address cross-cutting concerns such as scalability. In order to understand the architectural requirements of using the Web of Data as an open corpus for recommender systems, we need to analyse the architectural requirements of applications using Semantic Web technologies. In order to reach this goal we have performed a survey of 98 applications using Semantic Web technologies. We will first discuss the methodology of the survey and define the term “Semantic Web application”. Then we present the Semantic Web application reference architecture which is the result of the survey. We conclude the section by discussing related surveys of Semantic Web technologies and applications.

8.3.1 Methodology of the survey

The survey of current Semantic Web applications has been performed in two parts, consisting of an architectural analysis and a questionnaire about the application functionality.

**Architectural analysis** The applications from two key demonstration challenges in the Semantic Web domain have been analysed to identify the most common functionality of Semantic Web applications: the “Semantic Web challenge”[^1], organised as part of the International Semantic Web Conference from 2003 to 2008, and the “Scripting for the Semantic Web challenge”[^2] organised as part of the European Semantic Web Conference from 2006 to 2008. Duplicate submissions have been eliminated, resulting in a total number of 98 surveyed applications.

The result of the architectural analysis is a list of components which provide an abstraction of the most common functionality which is required to implement Semantic Web standards. The components have been extracted from the architecture diagrams and the textual descriptions of the application architecture and implementation, depending on availability in the submitted paper. The components provide a common way to decompose the surveyed applications, so that components with similar functionality from different applications can be compared. This allows us to e.g. identify the need for data updating standards, as most applications have a user interface, but only a minority of applications allow creation of new data by the user.

**Application functionality questionnaire** Additionally a questionnaire was used to collect details about the implementation of the applications. The questionnaire contains 27 properties associated with 7 areas of functionality. The results from the questionnaire provide statistics about the range of variations in which the functionality of the common components has been implemented.

The questionnaire covers these areas of functionality: (1) implementation of Semantic Web standards, (2) support for data sources, (3) support for formal vocabularies that are heterogeneous and have diverse ownership, (4) implementation of data integration and alignment, (5) support for structured, semi-structured, unstructured or multimedia data, (6) support for authoring and editing of data, and (7) support for external data sources and the open-world assumption.

Only the applications from the “Semantic Web challenge” 2003 to 2006, and the “Scripting for the Semantic Web challenge” 2005 to 2007 were analysed with the questionnaire. The authors of the papers describing the applications where asked to validate and correct the details about their applications. Of the 50 applications analysed with the questionnaire, 74% validated their data.

8.3.2 The survey result: the reference architecture

The two parts of the survey can be combined to provide an overview of the state of the art in implementing the required functionality for Semantic Web technologies. The architectural analysis provides a list of the

[^1]: http://challenge.semanticweb.org/
[^2]: http://www.semanticscripting.org
most common components, and the questionnaire provides statistical data about the different variations of implementing each component. Table 1 shows the seven most common components, and lists the number of applications implementing a specific component by year.

<table>
<thead>
<tr>
<th>year</th>
<th>number of applications</th>
<th>data interface</th>
<th>persistence storage</th>
<th>user interface</th>
<th>integration service</th>
<th>search service</th>
<th>authoring interface</th>
<th>crawler</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003</td>
<td>10</td>
<td>100%</td>
<td>80%</td>
<td>90%</td>
<td>90%</td>
<td>80%</td>
<td>20%</td>
<td>50%</td>
</tr>
<tr>
<td>2004</td>
<td>16</td>
<td>100%</td>
<td>94%</td>
<td>100%</td>
<td>50%</td>
<td>88%</td>
<td>38%</td>
<td>25%</td>
</tr>
<tr>
<td>2005</td>
<td>6</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>83%</td>
<td>83%</td>
<td>33%</td>
<td>33%</td>
</tr>
<tr>
<td>2006</td>
<td>19</td>
<td>100%</td>
<td>95%</td>
<td>89%</td>
<td>63%</td>
<td>68%</td>
<td>37%</td>
<td>16%</td>
</tr>
<tr>
<td>2007</td>
<td>24</td>
<td>100%</td>
<td>92%</td>
<td>96%</td>
<td>88%</td>
<td>88%</td>
<td>33%</td>
<td>54%</td>
</tr>
<tr>
<td>2008</td>
<td>23</td>
<td>100%</td>
<td>87%</td>
<td>83%</td>
<td>70%</td>
<td>78%</td>
<td>26%</td>
<td>30%</td>
</tr>
<tr>
<td>total</td>
<td>98</td>
<td>100%</td>
<td>91%</td>
<td>92%</td>
<td>72%</td>
<td>81%</td>
<td>32%</td>
<td>35%</td>
</tr>
</tbody>
</table>

Table 1: Percentage of surveyed applications implementing the 7 most common components, per year and in total

To define our reference architecture, we abstract from the empirical overview of existing systems, a summary of which was shown in [1]. We abstract from the differences between these systems and distinguish seven main components that can be clearly observed to exist in the majority of Semantic Web applications. For each component we will suggest a name, list other common names, give a description of the role and of distinguishing features, and then list the most common variation points of the component found among the surveyed applications.

The surveyed applications share a significant amount of functionality regarding common capabilities of Semantic Web applications. We abstract from the differences between individual applications and distinguish seven main components, which together constitute a reference architecture for Semantic Web applications by describing high-level concepts and terminology, without fixing interfaces [25] (page 242).

The (i) **data interface** provides an abstraction over remote and local data sources, the (ii) **persistent storage** stores data and run time state, and the (iii) **user interface** provides access for the user. (i) to (iii) have each been implemented by more than 90% of surveyed applications. The (iv) **integration service** provides a unified view on heterogeneous data, and the (v) **search service** allows searching in data. (iv) and (v) have each been implemented by 70% to 80% of surveyed applications. The (vi) **crawler** discovers and retrieves remote data, and the (vii) **authoring interface** allows creating new data and editing existing data. (vi) and (vii) have each been implemented by 30% to 40% of surveyed applications.

In the following we describe the functionality of each component in detail and provide statistical data for the range of variations amongst the implementations of the surveyed applications. The full results of the architectural analysis are available on-line [4] as are the details of the questionnaire results [5].

**Data Interface:** Also known as data adapter or data access provider. Provides the interface needed by the application logic to access local or remote data sources, with the distinction based on either physical remoteness or administrative and organisational remoteness. Separation from the persistence layer is moti-
Component variations: Accessing local data is implemented via programmatic access through RDF libraries by at least 50% of the applications. Only 24% use a query language for accessing local or remote data sources, but only half of these applications use the SPARQL standard. Multiple data sources with different ownership are used by 90% of applications, 70% support external data provided by the user and 60% can export their data or make it reusable as a source for other applications, by e.g. providing a SPARQL end-point. 76% of the applications support updating their data during application runtime.

Persistent Storage: Also known as persistence layer or triple store. Provides persistent storage for data and run time state of the application, it is accessed via the data interface. In practice many triple stores and RDF libraries provide both a data interface and persistent storage, but there are cases where the components are de-coupled, e.g. if the application has no local data storage, and only uses SPARQL to access remote data. 91% have a persistent storage.

Component variations: Possible supported standards include but are not limited to data representation languages (XML, RDF), meta-modelling languages (OWL, RDFS) and query languages (SQL, SPARQL). RDF is explicitly mentioned by 86% of applications, OWL is supported by 48%, RDFS by 22%. Inferencing or reasoning on the stored data is explicitly mentioned by 58% of the applications. Storage of any combination of structured, semi-structured, unstructured data or (binary) files can be implemented, with different levels of features or optimisation for the different data types. 58% implement support for unstructured text and 48% support mixing of structured and unstructured data in some way.

User Interface: Also known as portal interface or view. Provides a human accessible interface for using the application and viewing the data. Does not provide any capabilities for modifying or creating new data. 92% have a user interface, as some applications do not provide a human usable interface.

Component variations: The navigation can be based on data or metadata, such as a dynamic menu or faceted navigation. The presentation may be in a generic format, e.g. in a table, or it may use a domain specific visualisation, e.g. on a map (10%). 16% present images to the user and 6% explicitly mention support for audio content in the user interface. 28% support multiple languages in the user interface, thus catering to a multilingual audience.

Integration Service: Also known as integration, aggregation, mediation, extraction layer or service. Provides means for addressing structural, syntactic or semantic heterogeneity of data, caused by accessing data from multiple data sources using diverse kinds of format, schema or structure. The desired result
is a **homogeneous view on all data** for the application. The integration service often needs to implement domain or application specific logic for the data integration. 72% of the applications have an integration service.

**Component variations:** Integration of heterogeneous data is supported by 90% of the applications, and 90% support data from sources with different ownership. Data from distributed data sources is supported by 72%. These three properties are orthogonal, as it would be e.g. possible to support just SIOC data [15] which is not heterogeneous, but which is aggregated from personal websites, so that the data sources are distributed and under different ownership. Mapping or alignment between different schema may be automatic (12%), but most applications (80%) require some form of human intervention for the integration. Reasoning and inferencing can be used for the integration (58%). Integration may be performed once if data stays static, or continuously if new data gets added.

**Search service:** Also known as query engine or query interface. Provides the ability to perform searches on the data based on the content, structure or domain specific features of the data. Interfaces for humans, machine agents or both can be provided. 81% provide a search service.

**Component variations:** Besides search on features of the data structure or semantics, generic full text search (58%) or a search on unstructured and structured data at the same time (48%) can be provided. The interface for machine agents may be provided by e.g. a SPARQL, web service or REST endpoint.

**Crawler:** Also known as harvester, scatter or spider. Required if data needs to be found and accessed in a domain specific way before it can be integrated. Implements automatic discovery and retrieval of data. 35% implement a crawler. Some applications have an integration service, but do not need a crawler, e.g. because they only use local RDF data, but need to perform object consolidation [39].

**Component variations:** Support of different discovery and access mechanisms, like HTTP, HTTPS, RSS. Natural language processing or expression matching to parse search results or other web pages can be employed. The crawler can be active once if data is assumed to be static or continuous (76%) if new data needs to be discovered.

**Authoring interface:** Allows the user to enter new data, edit existing data, and import or export data. This component depends on the user interface component, and enhances it with capabilities for modifying and writing data. Separation between the user interface and the authoring interface reflects the low number of applications (32%) implementing write access to data.

**Component variations:** The annotation task can be supported by a dynamic interface based on schema, content or structure of data. Direct editing of data using standards such as e.g. RDF, RDF Schema, OWL or XML can be supported. Input of weakly structured text, using e.g. wiki formatting can be implemented. Suggestions for the user can be based on vocabulary or the structure of the data.

### 8.3.3 Related work for the reference architecture

Our methodology is adapted from [29], which uses six cases studies of software systems as the basis for introducing the basic concepts of software architecture. This is used as the foundation for identifying the most important general challenges for the design of complex software systems which are constructed from many components. We adapt this approach for the field of Semantic Web technologies. The challenges which we identify are based on a survey of 98 Semantic Web applications.

Other empirical surveys about Semantic Web applications are publicly available, however they are not concerned with the software architecture and specific implementation details of concrete applications. Thus
they provide no empirical basis for identifying the main challenges of implementing Semantic Web technologies. [20] presents the results of a survey of 627 Semantic Web researchers and practitioners done in January 2007. The questions from the survey cover the categories of demographics, tools, languages and ontologies. It tries to characterise the uptake of Semantic Web technologies and the types of use cases for which they are deployed. Another similar survey of 161 researchers and 96 application-oriented participants was published online in 2009.[7]

[22] performs a survey and architectural analysis of 35 applications from the “Semantic Web challenges” in 2003, 2004 and 2005. The result is a prescriptive software architecture for Semantic Web applications described with UML. However, the results of the survey do not identify any software engineering challenges for implementing Semantic Web technologies.

While no other empirical analysis of the challenges of implementing Semantic Web applications exist, the ONTOCOM project [68] provides a detailed cost estimation model for ontology development projects. We do not provide a cost estimate model for software engineering of Semantic Web applications. However, our identification of the main challenges in implementing such applications provides the basis for future research on establishing such cost estimation models.

8.4 Step 3: Applying the reference architecture to open recommenders

Our third step and final step is to specify how the architecture and components of open applications on the Web of Data can be applied to recommender systems on the Web of Data in order to provide a scalable and open architecture. We explain how each component from the Semantic Web reference architecture can be used as part of an open recommender.

Figure 8 shows the architecture of the music recommendation application from section 7. This open recommender system contains components from the Semantic Web reference architecture in order to be able to access the Web of Data. This example does not include all of the components from the reference architecture, however each of those components might be used in an open recommender system. In the following we explain how the components of the reference architecture might contribute to the functionality of an open recommender:

The data interface provides the application programming interface to access data sources from the Web of Data. It should always support the basic standards of the Web of Data such as HTTP, RDF and SPARQL (see section 4.2). The example application uses this to access DBpedia and DBtunes.

The crawler is used to discover data sources which contain data for the recommendation. Sources might be relevant because they contain data using a specific vocabulary such as FOAF or SIOC, or in the case of our example musical artists. The crawler uses the data interface to find this data, and then passes the data the integration service.

The integration service is used to provide a homogeneous view on the data of all accessed data sources. In our example we use SPARQL to integrate data which uses the four schema and vocabularies of FOAF, the MySpace Vocabulary, the Music Ontology and the SIOC ontology. The integrated data only uses the FOAF vocabulary, and is passed to the persistent storage.

The persistent storage is used to store the integrated data so that the integration does not have to be performed again after restarting the recommender system. It also acts as a cache to speed up access to

6 http://preview.tinyurl.com/semweb-company-austria-survey
data, as accessing remote data is slower then accessing local data. For our music recommender, the persistent storage is used to store the acquired and integrated background data, so that it can be used by the recommendation algorithm.

The **user interface** can be a generic user interface or it can provide a user interface which takes the structured nature of the data into account by providing e.g. faceted navigation. For a recommender system, the user interface is also used to collect the input data for an individual user, which is then used by the recommendation algorithm to provide the personalisation.

The **search service** is used by the user or by other Semantic Web applications to query and search for items. As part of a recommender system the results of the search service can be personalised based on the input data of a specific user and the background data of the recommender system. The search service is not used in the music recommender example.

The **authoring interface** is used to create new items or to edit existing items. As part of a recommender system, recommendations might be provided while creating or editing the properties of an item, e.g. when creating a band for a specific label, then similar bands might be recommended. The authoring interface is not used in the music recommender example.

By using the components from the Semantic Web reference architecture, we can augment the architecture of a recommender system, in order to enable it to use the Web of Data for open and scalable personalisation.
9 Conclusion

Recommendation services which only use their own data are based on the assumption of a closed architecture: the data is collected in-house for the purpose of providing recommendations for a specific domain. The amount of available data depends on the duration of collecting data. The requirements of the used recommendation algorithm in terms of data features can be taken into account before starting to collect data. If data needs to be integrated, all schemas are known in advance. The effort of cleaning up data is low because all sources are known.

An approach for new service providers to compete with existing recommendation services, is to make use of publicly available data sources. However, using open data sources for adaptive personalisation introduces new challenges for recommendation services. Such recommendation services need to be based on the assumption of an open architecture: data for the recommendation has to be collected from different sources. The amount of available data depends on the number and size of suitable data sources. The recommendation algorithm has to be chosen based on the features of the available data. Since the features of the data are not known in advance, feature selection plays an important role. Data has to be integrated without prior knowledge of the schema used by different sources. Different sources can add impurities to the data in unexpected ways. Finally, the different steps of the recommendation process might be performed by different parties.

In this report we have shown that the Web of Data can be used as an open corpus for adaptive personalisation and recommendation. It provides data and knowledge which can be used for established recommendation algorithms. In addition it provides links between related topics from different domains and identical users from different communities. These intrinsic properties of Linked Data have allowed us to propose new use cases for adaptive personalisation.

We have addressed all the identified main challenges related to the architectural design of recommendation and personalisation services and their use of an open corpus to a certain degree:

1. we have shown that the data acquisition problem can be addressed for a collaborative filtering algorithm by using data from DBtune and DBpedia, and have implemented and evaluated a prototype
2. we have discussed possible sources of formal knowledge and use cases for exploiting it, in order to address the knowledge acquisition problem
3. we have discussed possible solution approaches for the open corpus problem through a use case and a detailed example
4. we have proposed an architecture for open and scalable recommender systems on the Web of Data which is derived from a reference architecture for Semantic Web applications. The reference architecture is based on an empirical analysis of 98 Semantic Web applications. We provided an example of applying this architecture to a recommender system.

Future work: By building on the research presented in this report, we will further explore the potential and challenges of opening up the potential of the Web of Data for adaptive personalisation as well as exploring approaches for building the next generation of systems for adaptive personalisation and recommendation.

We will also put a special focus on using distributed case based reasoning to address the issues of selecting relevant data sources and routing queries for data to the right sources in order to use external data for adaptive personalisation. We will provide an implementation of the use case for using case-based
reasoning for expert finding. FOAF profiles could provide this information, and there are opportunities to collaborate with other DERI researchers on this.

References


