An Open Framework for Multi-source, Cross-domain Personalisation with Semantic Interest Graphs

Benjamin Heitmann

Ph.D. Thesis

Supervisor: Dr. Conor Hayes

External Examiner	**Internal Examiner**
Dr. Tommaso Di Noia	Dr. John Breslin

Insight Centre for Data Analytics
College of Engineering and Informatics
National University of Ireland, Galway

June 2014

Abstract

The work in this thesis addresses new challenges and opportunities for online personalisation posed by the emergence of new infrastructures for sharing user preferences and for access to open repositories of data. As a result of these new infrastructures, user profiles can now include data from multiple sources about preferences in multiple domains. This new kind of user profile data requires a cross-domain personalisation approach. However, current cross-domain personalisation approaches are restricted to proprietary social networking ecosystems.

The main problem that we address in this thesis, is to enable cross-domain recommendations without the use of a proprietary and closed infrastructure. Towards this goal, we propose an open framework for cross-domain personalisation. Our framework consists of two parts: a conceptual architecture for recommender systems, and our cross-domain personalisation approach. The main enabling technology for our framework is Linked Open Data, as it provides a common data presentation for user preferences and cross-domain links between concepts from many different domains.

As part of our framework, we first propose a conceptual architecture for Linked Open Data recommender systems that provides guidelines and best practices for the typical high level components required for providing personalisation in open ecosystems using Linked Open Data. The architecture has a strong empirical founding, as it based on an empirical survey of 124 RDF-based applications.

Then we introduce and thoroughly evaluate SemStim, an unsupervised, graph-based algorithm for cross-domain personalisation. It leverages multi-source, domain-neutral user profiles and the semantic network of DBpedia in order to generate recommendations for different source and target domains. The results of our evaluation show that SemStim is able to provide cross-domain recommendations, without any overlap between target and source domains and without using any ratings in the target domain.

We show how we instantiate our proposed conceptual architecture for a prototype implementation that is the outcome of the ADVANSSE collaboration project with CISCO Galway. The prototype shows how to implement our framework for a real-world use case and data.

Our open framework for cross-domain personalisation provides an alternative to existing proprietary cross-domain personalisation approaches. As such, it opens up the potential for novel and innovative personalised services without the risk of user lock-in and data silos.

Declaration

I declare that this thesis is composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification.

Galway, June 2014

Benjamin Heitmann

Acknowledgements

This PhD thesis would not have been possible without the help and support of many people.

First I would like to thank my thesis supervisor, Dr. Conor Hayes, for his sound advice, insightful feedback and almost limitless support during my PhD studies. Furthermore, I would like to thank my examiners, Dr. Tommaso Di Noia and Dr. John Breslin, for their suggestions and their feedback. Then I would like to thank Maciej Dabrowski, Richard Cyganiak and Alexandre Passant for fruitful research collaborations and Stefan Decker and Manfred Hauswirth for their scientific advice. I thank Mario Arias for developing HDT, which was a crucial piece of infrastructure for my research. I would also like to thank Hilda, Claire, Christiane and Carmel for their help in all administrative things.

Then I would like to thank everybody who worked with me during my PhD research, at INSIGHT @ NUI Galway, formerly known as the Digital Enterprise Research Institute (DERI), as well as all my friends whom I meet in Ireland. As is the case for many people in our post-modern times, there is a big overlap between my friends and co-workers. In particular, I would like to thank the following people: Ioana, Vaclav, Marcel, Erik, Hugo, John, Sam and Donn from the Unit of Information Mining and Retrieval (UIMR), for their feedback and support; Then in no particular order and without being complete, Laura, Alex, Josi, Jürgen, Maciej, Anna, Gregor, Sabrina, Jacek, Iza, Richard, Sheila, Knud, Renaud, Georgetta, Gabi, Laleh, Brian, Alessandra, Annelie, Myriam, Vit, Hanka, Lukasz, Pierre, Behrang, Fadi, Gofran, Brahmananda, Eyal for their support and for all the craic over the years. As well as Ronan, Mark, Sean, Jonny, Niall, Jude, Mary, James and James for the games and more craic. I thank Niall O'Floinn for teaching Chen Tai Chi at a world class standard in Galway.

Finally, I would like to thank my "support network" back in Germany, my friends Pascal, Reinhard, Johannes, and my family, Christian, Jana, Vera, Martina and Manfred.

The research presented in this thesis has been supported by:

- Science Foundation Ireland under Grant Number SFI/08/CE/I1380 (Lìon-2).
- Science Foundation Ireland under Grant Number SFI/12/RC/2289 (INSIGHT).
- the ADVANSSE project funded by Cisco Systems.

Contents

"The future is already here. It is just not very evenly distributed."
— William Gibson

Chapter 1.

Introduction

The work in this thesis addresses new challenges and opportunities for online personalisation posed by the emergence of new infrastructures for sharing user preferences and for access to open repositories of data. These new infrastructures introduce changes to the architecture of personalised services. As a result, recommender systems have shifted from closed inventories to open inventories. This has enabled the emergence of ecosystems, which provide the infrastructure to share user preference data between an external 3rd party service and the central hub of the ecosystem. Facebook[1] is an example of a hub site in such an ecosystem.

The changes in the architecture of recommender systems are accompanied by changes in the type of data available for profiling user preferences. User profiles can now include data from multiple sources about preferences in multiple domains. As we will show, a cross-domain personalisation approach is required in order to leverage this new kind of user profile data.

Cross-domain recommendation is a novel personalisation approach. It has the potential to enable personalised online experiences even if no preference data about the items in a recommender system are available for a user. For instance, an online travel agency can recommend travel destinations using the music preferences of a user. However, current ecosystems in which cross-domain personalisation is employed are fundamentally closed.

The main problem, which we address in this thesis, is to enable cross-domain recommendations without the use of a proprietary and closed infrastructure. Towards this goal, we present an open framework for cross-domain personalisation using semantic interest graphs. Our framework consists of a conceptual architecture and a personalisation approach. The proposed conceptual architecture describes how to leverage Linked Open Data for recommender systems. The proposed personalisation approach describes how our cross-domain recommendation algorithm can use Linked Open Data to generate cross-domain recommendations.

[1] https://www.facebook.com/

Our open framework for cross-domain personalisation provides an alternative to existing proprietary cross-domain approaches. As such, it opens up the potential for novel and innovative personalised services without the risk of user lock-in and data silos.

1.1. Motivation

Today, many of the most popular web sites on the World Wide Web feature personalised content of some form. According to marketing surveys[2] from May 2013, 75% of consumers prefer that retailers use personal information to improve their online experience. Complementary to this, 94% of companies agree that online personalisation is critical to the performance of their business. This suggests that personalisation has become an ubiquitous and expected feature for both consumers and companies on today's Web.

Some of the most prominent examples of personalised web sites, include e-commerce sites, such as Amazon[3], media discovery services, such as Netflix[4] and Spotify[5], and social networking sites, such as Facebook and Google Plus[6].

E-commerce sites use personalisation to provide each user with a personalised shopping experience, by e.g. suggesting items which are similar to the one which a user is currently viewing, or by suggesting items which are often bought together. This benefits users by allowing them to explore more of the inventory, and it benefits the company operating the site, as they might have a higher turn-over of sold items.

Media discovery sites recommend movies or music for immediate viewing or listening, based on a users previous preferences. This benefits users by allowing them to find more content which is relevant for them, while the media discovery site can increase user satisfaction which in turn can increase user retention.

Social networking sites use personalisation to select the most "important" content generated by the friends of a user. In this way, users can see the most important and interesting content generated by their friends, without being overwhelmed by content which they would perceive as uninteresting.

The changing architecture of recommender systems

For these three types of web sites (e-commerce, media discovery and social networking sites), personalisation is essential to the user experience. However, while all of these

[2] http://venturebeat.com/2013/05/07/mass-marketing-vs-personalization-infographic/
[3] http://www.amazon.co.uk/
[4] https://www.netflix.com
[5] https://www.spotify.com/
[6] https://plus.google.com

web sites offer superficially similar personalisation services to their users, the recommender systems that enable the personalisation are based on two fundamentally different assumptions regarding the origin of recommendable items:

- Recommender systems for e-commerce sites and media discovery sites are based on the assumption of a *closed inventory* (also called a "closed corpus"). For e-commerce sites, the items in the inventory of the service represent the items available in a physical warehouse. For media discovery services, the items in the inventory represent the items that are licensed for transmission to the user. The inventory is closed, as new items can only be added by the service operator.
- Recommender systems for social networking sites however are based on the assumption of an *open inventory* (also called an "open corpus"). The reason for this, is that the items in the inventory represent user generated content which has been created either on the site itself, or which links to content on an external site. The inventory needs to be open, as the items in the inventory can be created by the operators of the site as well as by the users of the site.

In addition, in recent years, many social networking sites have also each become the centre of an *ecosystem*, which provides the infrastructure for *sharing of user preferences* between external 3rd party services and the central hub site. For instance, users can connect the Spotify music discovery service to their Facebook user account. This allows Spotify to automatically share every song played by the user to his or her Facebook profile. In this way, Spotify as an external 3rd party service can add music preferences of the user to his or her Facebook profile. Such social networking ecosystems can increase the perceived value of the social network for the user and result in stronger user attachment.

Both the shift to open inventories and the emergence of ecosystems for sharing of user preferences, lead to recommender systems with a very different architecture, compared to closed corpus recommender systems.

New requirements for recommendation algorithms

The changes in the architecture of recommender systems are accompanied by changes in the type of data available for profiling user preferences. This in turn introduces new requirements for the recommendation algorithm.

The available user preference data consists of user profiles which are (a) aggregated from different services (*multi-source profiles*), and (b) not associated with a specific domain or inventory (*domain-neutral profiles*) [Abel et al., 2011]. In order to use this new type of user profile data for personalisation, a recommendation algorithm is required that can use preferences from any domain in a multi-source user profile to recommend items in the inventory of a personalised service.

Cross-domain recommendation algorithms allow items from one domain to be recommended based on preferences in another domain [Fernández-Tobías et al., 2012]. For instance, travel destinations can be recommended, based on the books which a user has read.

By employing a cross-domain recommendation algorithm, personalised services are enabled to provide recommendations for *new users* without any preferences about items from that particular service. The problem of new users without any preferences is a part of the cold start problem [Schein et al., 2002].

As an example of the new-user problem, consider a user who is visiting a restaurant recommendation service for the first time, and who does not have any food preferences in his or her user profile. Using a cross-domain recommendation algorithm allows the restaurant recommendation service to recommend restaurants to the new user, by using the preferences from other services which have been aggregated in the multi-source profile of the user. For instance, the restaurant recommendation service could use the news articles that a user has read in order to recommend restaurants to the new user.

Using Linked Open Data to provide an open framework for cross-domain personalisation

The main problem, which we address in this thesis, is to enable cross-domain recommendations without the use of a proprietary and closed infrastructure.

Current social networking ecosystems are fundamentally *closed*, in order to give the user a sense of privacy. The user model and all aggregated data about a user belong to the social network operator. This is meant to protect the privacy of the user when his or her profile data is exchanged between the hub site and an external 3rd party service. However, it also creates user lock-in and data silos [Chellappa & Sin, 2005a].

We approach the problem of providing an alternative and open framework for cross-domain personalisation in two stages:

- Firstly, we propose a conceptual architecture that describes how to share and integrate user preference data between personalised services.
- Secondly, we propose a personalisation approach that can use the obtained multi-source user profile data to provide cross-domain recommendations.

In our thesis, we derive the requirements for our cross-domain personalisation approach from the requirements of personalisation in current social networking ecosystems. These requirements are related to the data representation of the user profiles:

1. The preferences of a single user can be distributed across multiple services in an ecosystem. Therefore, a data representation format is required that allows for the aggregation and integration of user profiles from multiple sources.

2. Each source of user preferences, which contributed to a multi-source user profile, can describe a different domain. Therefore a data representation format is required that allows for merging of user preferences from multiple domains.

Both of these requirements depend on a common data representation of user preferences, which is used by all personalised services in an ecosystem.

We argue that these two data representation requirements are fulfilled by the availability of Linked Open Data [Bizer et al., 2009] as an enabling technology for our proposed cross-domain personalisation framework. Linked Open Data is suitable to enable cross-domain personalisation for three reasons:

1. The Linking Open Data cloud provides data sets from many different conceptual areas which have been published by many different data providers using RDF and the Linked Data principles. These data sets provide *re-usable concept identifiers*, in the form of RDF URIs.

2. The Linked Data principles and the RDF data model specify a standardised, open and interoperable way for *expressing graph data*.

3. In addition, the Linking Open Data cloud provides *cross-domain links between concepts from many different conceptual areas*, as the Linked Data principles explicitly encourage data providers to publish links between concepts from their Linked Data sets and concepts from other Linked Data sets.

Conceptual architecture for Linked Open Data recommender systems

Our framework provides a conceptual architecture for open recommender systems which use Linked Open Data in order to enable the sharing of user preference data.

In order to provide a strong empirical grounding for this conceptual architecture, we performed an empirical survey of 124 RDF-based applications. The applications come from the years 2003 to 2009 of the "Semantic Web challenge" and the years 2006 to 2009 of the "Scripting for the Semantic Web challenge". These are two key academic demonstration challenges in the Semantic Web research community. This allows us to look back on the empirical evidence from RDF-based applications that have been developed and deployed during this time.

The proposed conceptual architecture for Linked Open Data recommender systems provides a template and best practices for the typical high level components required for providing personalisation in open ecosystems. Prospective developers can use it

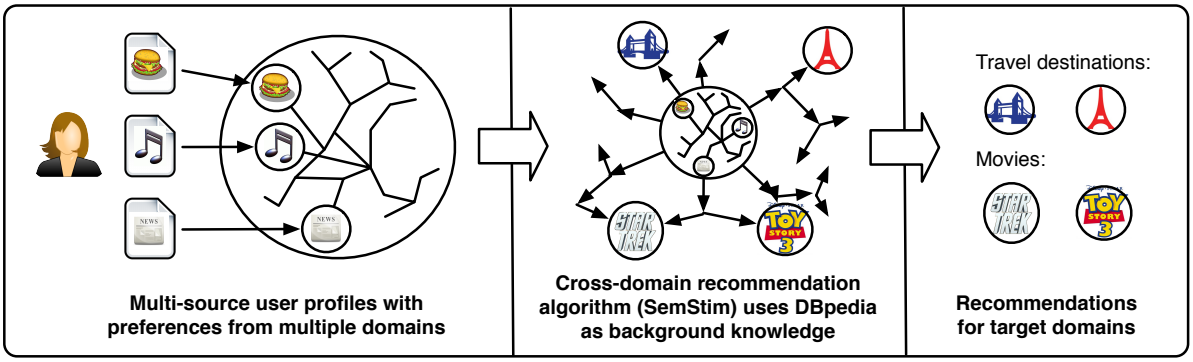

Figure 1.1.: Overview of our cross-domain personalisation approach

as a guideline when designing and implementing a recommender system which uses Linked Open Data to enable cross-domain recommendations using multi-source and domain-neutral user profiles.

Personalisation approach for cross-domain personalisation

In addition to the conceptual architecture, our framework provides a personalisation approach for cross-domain personalisation, which is based on SemStim, an unsupervised, graph-based algorithm for cross-domain personalisation. Figure 1.1 shows a high-level overview of our personalisation approach.

SemStim extends the spreading activation (SA) algorithm as described by [Crestani, 1997] for the purpose of cross-domain recommendation. We add (1) *targeted activation*, which allows us to describe the target domain of the personalisation, and (2) we add a set of *constraints* to control the algorithm *duration*.

SemStim uses inter-domain links from DBpedia to find indirect connections between items from different domains. This enables SemStim to recommend items from one domain based on user preferences in another domain.

We evaluate SemStim, by comparing it to two other graph-based algorithms, Linked Data Semantic Distance (LDSD) and set-based breadth-first search (SetBFS). In addition, we compare SemStim to k-nearest neighbor collaborative filtering (CFknn), and SVD++, which is a collaborative filtering algorithm based on matrix factorisation.

We describe the results of three experiments. First we compare the single-domain accuracy of all algorithms. Then we compare the accuracy of cross-domain recommendations generated by SemStim and the other two graph algorithms. In addition, we evaluate the diversity of the recommendations generated by the different algorithms.

Finally, we present the results of our participation with SemStim at the Linked Open Data-enabled Recommender Systems (LODRecSys) challenge at the Extended Semantic

Web Conference (ESWC) 2014. The results show, that SemStim has a competitive performance for the single-domain diversity recommendation task, as we were ranked on 3rd place out of 12 participants for the diversity recommendation task.

ADVANSSE prototype implementation

We describe a prototype implementation, which is the outcome of the ADVANSSE[7] collaboration project with CISCO Galway. The ADVANSSE prototype is based on our cross-domain personalisation framework, and it shows how to implement all parts of our framework for cross-domain personalisation for a real-world use case and for real-world data.

The ADVANSSE use case is based on different scenarios for knowledge sharing and discovery in a distributed enterprise environment. We describe all the details for the data used in the prototype, as well as for the implementation of the different parts of the ADVANSSE distributed social platform.

1.2. Research questions

In this thesis, we address the following research questions, based on the topics described in the previous sections:

Q1: *How can we mitigate the cold-start problem in order to provide recommendations for new users?*

For current state-of-the-art recommender systems, the new item and sparse data problems are often mitigated by combining collaborative filtering with content-based recommendation in a hybrid system [Burke, 2002], which allows identifying item similarity by using content features and/or meta-data. While hybrid recommender systems can mitigate the challenge of sparse data and new items, they are *not* able to address the problem of a new user without any known preferences for items in the recommender system itself. We will show that the new user problem can be addressed by using cross-domain recommender systems, if additional user preferences about items outside of the recommender system are available.

Q2: *How can we enable an open ecosystem for cross-domain recommendations outside of existing proprietary ecosystems?*

Today's social networking sites, such as Facebook or Google Plus provide the infrastructure for ecosystems which enable the sharing of user profile data between external 3rd-party personalised services and the hub site of the ecosystem. However, these ecosys-

[7] http://advansse.deri.ie/

tems are controlled by the social network operator. In addition, all user profile data belongs to the social network operator. We will show how Linked Open Data can enable alternative open ecosystems for personalised services.

Q3: *What kind of data structure should be used for domain-neutral user profiles, in order to allow merging profile data from multiple personalisation services?*

In order to participate in an open ecosystem for personalised services, multi-source and domain-neutral user profile data needs to be shared and aggregated between the services, after which it needs to be integrated. We will show how Linked Open Data can be used to enable aggregating and integrating of user profile data.

Q4: *Which recommendation algorithm can generate cross-domain recommendations, without any overlap between source and target domain and without any ratings for the target domain?*

Cross-domain recommender systems, as defined by Fernandez-Tobias et al. [Fernández-Tobías et al., 2012], use preferences from one or more source domains in order to generate recommendations in one or more target domains. Existing approaches for cross-domain recommendations either require an overlap between users or items of two domains, or they require ratings for the target domain. We determine and evaluate an algorithm that is able to provide cross-domain recommendations *without any* user preferences for the target domains, and without any overlap between source and target domain.

Q5: *Which components and best practices need to be implemented by a recommender system in order to leverage Linked Data for recommendations?*

We propose to use Semantic Web technologies and Linked Open Data (LOD) to provide the supporting infrastructure for personalisation in open ecosystems. However, there are no established best practices and guidelines for implementing the functionality which is required to leverage Linked Open Data. We will review the best practices for implementing LOD-based applications, with an additional focus on the functionality required for recommender systems. Based on this, we will present a conceptual architecture for recommender systems using Linked Open Data.

1.3. Contributions of the thesis

The main contribution of this thesis is a framework which provides cross-domain recommendations using Linked Open Data. Our framework addresses new requirements for personalisation approaches which arise from two recent architectural changes to recommender systems: (1) the shift of recommender systems from closed to open inventories, and (2) the emergence of ecosystems, which provide the infrastructure for sharing of

user profile data between personalised services. Our framework facilitates cross-domain personalisation outside of existing proprietary approaches.

1.3.1. List of main contributions

Further, through the individual parts of our framework, we make the following contributions:

Conceptual architecture for Linked Open Data recommender systems

We proposed a conceptual architecture for Linked Open Data recommender systems. Our conceptual architecture is novel, as it is based on a strong grounding of an empirical survey performed on 124 RDF-based applications from two key academic demonstration challenges over most of the past decade. Prospective developers can use it as a guideline when designing and implementing a recommender system which uses Linked Open Data to enable cross-domain recommendations using multi-source and domain-neutral user profiles.

Contributions:

- Our proposed conceptual architecture describes best practices and guidelines for leveraging Linked Open Data for recommender systems.
- We provide a list of the high-level components that a recommender system can implement in order to share, aggregate and integrate user profiles using Linked Open Data.
- We present the results of an empirical survey of 124 RDF-based applications from two key academic demonstration challenges.

Cross-domain recommendation algorithm: SemStim

We introduce and thoroughly evaluate SemStim, an unsupervised, graph-based algorithm for cross-domain personalisation.

SemStim is novel in the way it solves the problems of cross-domain recommendations as defined by [Fernández-Tobías et al., 2012] and [Cremonesi et al., 2011]. It uses indirect connections from DBpedia to generate recommendations between items of two different domains even in the worst case scenario when there is no overlap between users and items of the two domains.

SemStim is also novel as it is not affected by the *sparsity of rating data* or by the *cold-start problem* (cf. [Schein et al., 2002]). SemStim only requires user preferences of

the user for whom recommendations are being generated. In other words, it will even work when there is only a single user in the system. No ratings for the target domain are required. As a result, SemStim does not require any overlap between the target and source domains, and it does not require any ratings for the target domain.

Contributions:

- We provide a thorough evaluation of SemStim, by performing three experiments in which we compare SemStim to two Collaborative Filtering implementations and two graph-based algorithms.
- We show that SemStim can provide cross-domain recommendations for different source and target domains, without any overlap between the domains, and without using any preferences from the target domain.
- We show that SemStim can provide single-domain recommendations for equal source and target domains.
- Our evaluation shows that SemStim can mitigate the cold-start problem for new users, by using additional user preference data to provide cross-domain recommendations.
- We show that SemStim has a competitive performance for the task of balancing diversity and accuracy, when compared to other recommendation algorithms at the Linked Open Data-enabled Recommender Systems challenge at the Extended Semantic Web Conference 2014.
- We propose a novel diversity metric for recommendations, which is based on estimating the number of clusters in a set of DBpedia URIs.
- We show that the recommendations of SemStim can be tuned between being more diverse or more similar.

ADVANSSE prototype

We describe a prototype implementation that is the outcome of the ADVANSSE[8] collaboration project with CISCO Galway. The ADVANSSE use case is based on different scenarios for knowledge sharing and discovery in a distributed enterprise environment. We describe all the details for the data used in the prototype, as well as for the implementation of the different parts of the ADVANSSE distributed social platform.

Our prototype implementation shows that the framework for cross-domain recommendation which we present in this thesis is well suited for practical application in current industry settings.

Contributions:

[8] http://advansse.deri.ie/

- Our prototype implementation shows how we use Linked Open Data to enable an ecosystem for cross-domain personalisation.
- We describe how we implemented domain-independent profiles for the prototype implementation.
- We describe how we deployed our SemStim algorithm on real-world data to generate cross-domain recommendations.
- The prototype shows how to instantiate our conceptual architecture for Linked Open Data recommender systems.

1.4. Structure of the thesis

In the following, we describe the structure of the thesis and we briefly introduce the topic of each chapter, along with the research questions it primarily addresses.

Chapter 2 In the next chapter, we expand on the motivation of our research, and we introduce the background which is required in order to derive the research questions. In particular, we introduce the limitations of current state-of-the-art personalisation approaches, then we introduce the requirements for personalisation approaches in open ecosystems for personalised services. In the last part of the chapter, we introduce Linked Open Data.

Chapter 3 We provide best practices and guidelines for leveraging Linked Open Data for recommender systems in Chapter 3. This chapter addresses research question (Q5). We present a conceptual architecture for recommender systems using Linked Open Data. The proposed conceptual architecture is derived from an empirical survey of RDF-based applications, and describes the components and their high-level functionality, which a recommender system can implement in order to leverage Linked Open Data.

Chapter 4 We introduce SemStim, our algorithm for cross-domain personalisation in Chapter 4. This chapter addresses research question (Q4). SemStim is an unsupervised, graph-based recommendation algorithm which uses Linked Data from DBpedia in order to provide recommendations in a target domain using preferences in a different source domain. SemStim does not require any overlap between target and source domain, and it does not require any ratings in the target domain.

Chapter 5 We evaluate SemStim in Chapter 5, by comparing it to two graph-based algorithms, and to two implementations of Collaborative Filtering. We demonstrate the ability of SemStim to provide accurate recommendations for equal source and target domains, as well as for different source and target domains. In addition, we describe two experiments related to the diversity of the recommendations generated

by SemStim. This chapter shows that SemStim does not require an overlap between target and source domain, or ratings in the target domain (Q4). The chapter also shows that SemStim can provide recommendations for new users if an additional source of preferences from a different domain is available (Q1).

Chapter 6 We describe a prototype implementation, which is the outcome of the ADVANSSE collaboration project with CISCO Galway, in Chapter 6. The prototype shows how we instantiated our proposed conceptual architecture for Linked Open Data recommender systems, and how we deployed SemStim for the prototype. In addition, it shows which infrastructure we implemented for the prototype to support user profile sharing for an open ecosystem of personalised services (Q2), and how we implemented multi-source, domain-neutral user profiles (Q3) for the prototype.

Chapter 7 In the last chapter, we conclude the thesis, discuss the limitations of our cross-domain personalisation framework, and we present several directions for future research, which have emerged out of our research.

1.5. List of publications

The following is a list of publications which have been published as part of the research presented in this thesis. In addition, each chapter which is based on an existing publication, is marked as such at the beginning of the chapter.

Book chapters:

- Benjamin Heitmann, Maciej Dabrowski, Conor Hayes, Keith Griffin, "Towards near real-time social recommendations for the enterprise", In Gloria Phillips-Wren, Liana Razmerita, Lakhmi C. Jain (ed.), *Innovations in Knowledge Management - The Impact of Social Media, Semantic Web and Cloud Computing*, Springer, 2014.
- Benjamin Heitmann, Richard Cyganiak, Conor Hayes, Stefan Decker, "Architecture of Linked Data Applications", In Andreas Harth, Katja Hose, Ralf Schenkel (ed.), *Linked Data Management: Principles and Techniques in the Series on Emerging Directions in Database Systems and Applications*, CRC Press, 2014.

Journal papers:

- Benjamin Heitmann, Richard Cyganiak, Conor Hayes, Stefan Decker, "An empirically-grounded conceptual architecture for applications on the Web of Data", In *IEEE Transactions on Systems, Man and Cybernetics, Part C - Applications and Reviews*, 2011.

Conference papers:

- Benjamin Heitmann, Maciej Dabrowski, Conor Hayes, Alexandre Passant, Keith Griffin, "Personalisation of Social Web Services in the Enterprise using Spreading Activation for Multi-source, Cross-domain Recommendations", In *Proceedings of the AAAI Spring Symposium on Intelligent Web Services Meet Social Computing*, 2012.
- Benjamin Heitmann, Conor Hayes, "Using Linked Data to build open, collaborative recommender systems", In *Proceedings of the AAAI Spring Symposium on Linked Data Meets Artificial Intelligence*, 2010.

Workshop papers:

- Benjamin Heitmann, "An Open Framework for Multi-source, Cross-domain Personalisation with Semantic Interest Graphs", In *Ph.D. Symposium at the ACM Recommender Systems Conference*, 2012.
- Benjamin Heitmann, James G. Kim, Alexandre Passant, Conor Hayes, Hong-Gee Kim, "An architecture for privacy-enabled user profile portability on the Web of Data", In *Proceedings of the International Workshop on Information Heterogeneity and Fusion at the ACM Recommender Systems Conference*, 2010.

Posters:

- Benjamin Heitmann, Conor Hayes, "SemStim at the Linked Open Data-enabled Recommender Systems 2014 challenge", In *Proceedings of the Extended Semantic Web Conference*, 2014.

Chapter 2.

Background*

Personalisation through recommendation plays an important role for the user experience of most user-facing applications, and as such it has become an expected feature. However there are many application scenarios in which the majority of current state-of-the-art personalisation approaches can not be used, as they require user preferences matching the conceptual domain of the application. In other words, in order to provide e.g. travel recommendations, most current approaches require a user profile with travel preferences.

Using preferences in one source domain to provide recommendations in another target domain, is called *cross-domain personalisation*. However, most state-of-the-art personalisation approaches currently have two major shortcomings related to the task of cross-domain personalisation. Firstly, current personalisation approaches can not generate recommendations for new users without any known preferences, which is known as part of the *cold-start problem*. Secondly, current personalisation approaches are inherently limited to one conceptual domain, which is the domain for which the recommender system has data about recommendable items and user preferences available.

We define a domain as any set of recommendable items, together with a set of users with preferences about the recommendable items. While this allows considering e.g. all the books on Amazon as a domain, it also allows more abstract domains, such as all spring-sale items on offer by a travel company. We provide our formal definition of a domain in Section 2.2.3.

*This chapter is based partially on previously published work:

1. Benjamin Heitmann, Conor Hayes, "Using Linked Data to build open, collaborative recommender systems", In *Proceedings of the AAAI Spring Symposium on Linked Data Meets Artificial Intelligence*, 2010.
2. Benjamin Heitmann, "An Open Framework for Multi-source, Cross-domain Personalisation with Semantic Interest Graphs", In *Ph.D. Symposium at the ACM Recommender Systems Conference*, 2012.
3. Benjamin Heitmann, James G. Kim, Alexandre Passant, Conor Hayes, Hong-Gee Kim, "An architecture for privacy-enabled user profile portability on the Web of Data", In *Proceedings of the International Workshop on Information Heterogeneity and Fusion at the ACM Recommender Systems Conference*, 2010.

In this thesis, we aim to show that a recommender system can provide personalisation results for users without preferences in the domain of the recommender system, if preferences for the user from one or more other domains are available. In addition, this can mitigate the problem of providing recommendations for new users without preferences in the domain of the recommender system.

We present an open framework for cross-domain personalisation, which makes this novel kind of recommendation capability available outside of closed social networking ecosystems, such as Facebook. In order to reach our goal of enabling cross-domain recommendations outside of closed ecosystems, we also need to consider the architecture of the infrastructure of these ecosystems. The infrastructure allows personalised services to share and aggregate user preferences from multiple services. As we will show in this chapter, personalised social networking ecosystems introduce multiple new requirements that make current recommendation algorithms unsuitable for personalisation in these ecosystems.

Chapter outline:

In this chapter, we will first introduce the state-of-the-art related to research on recommender systems, as well as the limitations of current personalisation approaches that allows us to formulate our research questions for this thesis. We will provide a classification of recommendation algorithms, and we will explain what kind of data are required for the different types of recommendation algorithms. Then we introduce the cold-start problem, which occurs when a recommender system does not have sufficient data to provide recommendations.

In the second part of this chapter, we will describe current social networking ecosystems, and the way in which they employ personalisation. Social networking ecosystems introduce new requirements for personalisation, namely (1) multi-source user profiles that contain (2) domain-neutral user preferences. Providing personalised services on these kind of user profiles requires (3) a cross-domain personalisation algorithm because personalised services will encounter users without preferences in their domain. We describe each of these new requirements in detail and list the related research questions.

The last part of this chapter will describe Linked Data and the Semantic Web. As we will show in our description of the cross-domain recommendation task, in order to use preferences in one or more source domains to provide recommendations in a target domain, some form of connection between the source and target domain is necessary. As we will show, Linked Data can provide the indirect links between different domains.

2.1. Recommender Systems

Personalised recommendations have proven themselves to greatly enhance the user experience of searching, exploring and finding new and interesting content. 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 [Xiao & Benbasat, 2007]. In addition, e-commerce web sites benefit from recommender systems by converting browsers into buyers, increasing the average order size and improving consumer loyalty[Schafer et al., 2007].

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 [Adomavicius & Tuzhilin, 2005].

In this section, we describe the state of the art regarding personalisation approaches, by providing a classification of recommendation algorithms and we describe the different kinds of data which each type of algorithm requires. Finally we introduce the cold-start problem, which occurs when a recommender system does not have sufficient data to provide recommendations, and which affects different recommendation algorithms differently.

2.1.1. Classification of Recommendation Algorithms

Recommender systems require three components to provide recommendations [Burke, 2002]: (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.

Different recommendation algorithms require different types of background and input data in order to provide recommendations. Current recommendation algorithms can be grouped in 5 classes according to [Bobadilla et al., 2013]:

1. Collaborative filtering
2. Demographic filtering
3. Content-based filtering
4. Knowledge-based filtering (as a special case of content-based filtering)
5. Hybrid filtering approaches

Table 2.1 shows an overview of the background data and input data which is used by the different algorithm classes.

	Background data	Input data	Algorithm
Collaborative filtering	ratings of items by users	rating vector of a user	pair-wise similiarity function between users or items
Demographic filtering	demographic information about users and their ratings of items	demographic information about a user	identify demographically similar users to the active user and extrapolate item ratings from their items
Content-based filtering	content features or meta-data	list of preferred features	feature based similarity search
Knowledge-based filtering	domain knowledge about items	knowledge of user needs and preferences	reasoning on user and item knowledge
Hybrid filtering	depends on chosen algorithms	depends on chosen algorithms	depends on chosen algorithms

Table 2.1.: Summary of background data, input data and algorithm used by the presented personalisation approaches

Collaborative filtering

Collaborative filtering (CF) [Su & Khoshgoftaar, 2009] [Candillier et al., 2007] [Schafer et al., 2007] [Herlocker et al., 2004] aggregates feedback for items from different users, and uses similarities between users and users or between items and items to provide recommendations.

CF algorithms require background data about the feedback which uses have provided for items, no other information about either the users or the items is required. The user feedback is commonly modeled in the form of binary or numerical ratings. The input data 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.

Collaborative filtering has been a widely explored research topic in the Recommender Systems domain since the early 90's. The GroupLens project [Resnick et al., 1994] produced one of the first implementations of a collaborative filtering algorithm in order to filter net news from the usenet. In the meantime, collaborative filtering algorithms have reached a high maturity and are currently used by most major e-commerce vendors. As an example, Netflix [Amatriain & Basilico, 2012] uses CF as part of its personalisation approach, and although details are not open to use, it is widely believed that Amazon.com also uses CF for its personalised recommendations.

We describe k-nearest neighbour collaborative filtering (CFknn) in more detail in Chapter 5, where we use CFknn as one of the baseline algorithms in the evaluation of our own approach.

Demographic filtering

Demographic recommender systems [Aimeur et al., 2006] [Porcel et al., 2012] aim to categorise the user based on personal attributes and make recommendations based on demographic classes. The demographic features used for this classification are e.g. sex, age, nationality, occupation or current location. The intuition behind demographic filtering, is that users in the same demographic class have similar interests. Demographic filtering aims to identify user-to-user similarity like the CF approaches, however using different data.

Content-based filtering

Content-based recommendation approaches [Salter & Antonopoulos, 2006] [Pazzani & Billsus, 2007] use 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 or tempo of the music piece, or derived from the meta-data of the items, e.g. author, title and genre. The input data needs to describe the user preferences in terms of content features. Both the background and the input data require the consistent description of content features in order to match the user preferences to the features of the content.

Content-based personalisation can be based on existing ratings of items performed by users, or more implicit connections between the user and items, such as visited locations or news articles read by the user.

Knowledge-based filtering

Knowledge-based recommendation [Amini et al., 2011] [Middleton et al., 2009] is a special case of content-based filtering, where knowledge about users preferences and item features is used in conjunction with content features. Knowledge-based filtering approaches aim to suggest items based on inferences about the users' needs and preferences. This requires background data that 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.

Later in this chapter, in Section 2.3.1, we will introduce the Web of Data, the Linked Data principles and Semantic Web technologies. These technologies enable the knowledge-engineering which is required for knowledge-based filtering.

Knowledge-based approaches are different from CF and content-based recommendation, as they have functional knowledge, e.g. about how a particular item meets a particular user need, and can therefore reason about the relationship between a need and a possible

recommendation. This also allows them to use data about the context of the user in order to make recommendations.

Hybrid filtering approaches

Hybrid algorithms [Porcel et al., 2012] [Barragáns-Martínez et al., 2010] 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 [Burke, 2002], 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 an algorithm of a different type, 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.

2.1.2. Experiment protocols for evaluating accuracy of recommender systems

The two most commonly used experiment protocols for evaluating and comparing the accuracy of recommender systems, are the rating prediction task and the top-k recommendation task, according to [Shani & Gunawardana, 2011].

Rating prediction: The rating prediction task involves using each algorithm to predict the rating of a list of unrated items based on the known preferences of a given active user. The predicted ratings are then compared to the real ratings in the ground truth by quantifying the error of the rating prediction. Metrics such as the round mean square error (RMSE) or the mean absolute error (MAE) are then used to quantify the error of the rating prediction for each algorithm.

Top-k recommendation: The top-k recommendation task involves ranking a list L of unrated items for the active user. Then the first k items of the ranked list are recommended to the user. Usually the number k of items to recommend is much smaller then the number $|L|$ of all recommendable items, so that $k < |L|$. Typically $0 < k \leq 20$. The recommended items are then compared with the list of withheld items in the test profile of the active user. This allows using metrics such as precision, recall and the F1-score to measure the accuracy of the recommendations.

The most cited publication in the recommender systems area about using the top-K recommendation task to evaluate recommender systems is published by [Cremonesi et al., 2010]. The authors propose an experiment protocol that allows using accuracy metrics such as precision and recall for evaluating rating predictions. The authors explain that while

many accuracy evaluations in the recommender systems community are based on the rating prediction task, the top-K recommendation task is more relevant for use cases with a large number of items, such as personalisation in E-commerce or social media.

2.1.3. The cold-start problem

Most current real-world recommender systems, such as those used by Netflix [Amatriain & Basilico, 2012] or Amazon [Linden et al., 2003b] employ collaborative filtering (CF). Collaborative filtering is affected by the cold-start problem, which is the inability to make reliable recommendations due to an initial lack of ratings. The cold-start problem can appear in three cases [Bobadilla et al., 2013]: for (a) new items, (b) new users and when (c) the background data of the recommender system is very sparse.

The *new item problem* [Schein et al., 2002] occurs, when a new item is added to the recommender system without any ratings. As it does not have any ratings, it can not be compared to other items, which leads to the item not being included in any recommendations. In turn, as the item is not presented to any users, the item can not accumulate any new ratings.

The *new user problem* [Schein et al., 2002] occurs, when a new user joins the recommender system. The new user did not provide any preferences in the form of ratings, and so there are no preferences to compare the new user to existing users or items. Even if the user is aware of his need to provide ratings in order to get recommendations, the recommender system usually will not be able to generate recommendations using the first few ratings of the user.

This leads us to the third problem, which is the *data sparsity* problem, which is also known as the new community problem in [Bobadilla et al., 2013]. If the number of ratings is low compared to the number of items in the background data then ineffective recommendations will be generated [Balabanovic & Shoham, 1997]. In order to provide effective and relevant recommendations, collaborative recommendation algorithms need a sufficient amount of ratings data, dependent on the specific algorithm implementation.

In order to mitigate the cold start problem, a common strategy is to turn to additional information sources. The additional data then is used to complement the ratings in the background data of the recommender systems. In order to achieve this, CF algorithms are often combined with content-based or knowledge-based recommendation in a hybrid system [Burke, 2002]. This allows circumventing the challenge of sparse data by using content-features or meta-data in addition to user ratings. In addition, this allows the similarity between items to be determined based on their content-features, which in turn can mitigate the new item problem.

Research question:

However, while using a Collaborative Filtering algorithm in a hybrid recommender system can mitigate the challenge of sparse data and new items, they are not able to address the problem of a new user without any known preferences. This brings us to the first research question of this thesis:

Research question (Q1): How can we mitigate the cold-start problem in order to provide recommendations for new users?

In this thesis we aim to provide an alternative approach to mitigate the cold-start problem for new users. As we will show in the next section, the new user problem is important for recommender systems in personalised social networking ecosystems. In such settings there is also additional data about a user's preferences available.

In order to solve (Q1), we will propose both a new way to access this additional background data, as well as a new algorithm for generating the recommendations. In Chapter 3, we describe a conceptual architecture for Linked Open Data recommender systems, which enables transporting and aggregating user preferences which are distributed in a personalised ecosystem. Then we describe SemStim, our personalisation algorithm in Chapter 4. As we will show, by providing cross-domain recommendations, SemStim is able to mitigate the cold-start problem for new users if additional data about their preferences from other personalised systems is available.

2.2. Recommender systems for social networking ecosystems

In this section, we derive the requirements for our cross-domain personalisation approach from the requirements of current social networking ecosystems.

Initially, the field in which personalisation was most successful, was on e-commerce sites like Amazon.com [Linden et al., 2003a]. This early success then also made it a defining feature for the user experience on many other kinds of web sites, from media discovery services like Netflix [Bennett & Lanning, 2007] to social networking providers, such as Facebook [Kincaid, 2010].

On social networks, like Facebook, Google+ and Twitter, personalised recommendations have proven themselves to greatly enhance the user experience of searching, exploring and prioritising new and interesting content. However all of these social networking sites are also each the centre of an ecosystem which allows external services and third party sites to enhance the user experience. Music streaming services, news papers or image

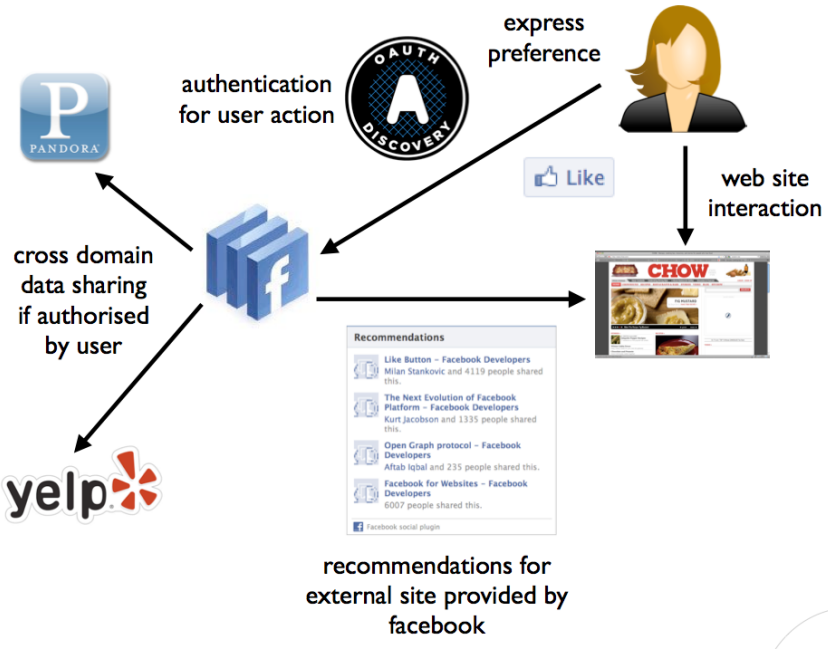

Figure 2.1.: Architecture of the Facebook social networking ecosystem

sharing sites can post to a users activity stream on behalf of the user, thus contributing interests to his profile.

In social networking ecosystems one site typically has the role of the *hub site*, which provides the main entry point for the whole ecosystem and stores the user profile data. Prominent social networking sites like Facebook, Google Plus and Twitter[1] are such central hub sites. *Third party services* can provide value-added and personalised services for the user of an ecosystem. Examples include the music streaming service Spotify, which can post all music a user listens back to his Facebook activity stream. The Guardian newspaper does the same for news articles read by the user. *Users* stay in control of their profile, as each user can specify which services have access to their user profile. This is summarised from our analysis of the developer sites of Facebook[2], Google Plus[3].

Figure 2.1 shows the architecture of the Facebook ecosystem, as an example of such a social networking ecosystem.

While this increases the value of the social network for the user and results in stronger user attachment, this also introduces completely new challenges for building recommender systems:

Multi-source user profiles: Each user of a hub site will have a user profile, with parts originating from multiple third party services. Each ecosystem must provide an

[1] http://www.twitter.com
[2] https://developers.facebook.com/docs/reference/login/open-graph-permissions/
[3] https://developers.google.com/+/api/

architecture and a transport mechanism for aggregating such a multi-source user profile from multiple third party services.

Domain-neutral user preferences: The user preferences of one user will not just be about items from one domain, such as movies. Instead the preferences will span the domains of multiple services which have contributed to the user profile, thus resulting in the user profile being domain-neutral.

Cross-domain recommendations: Third party services which want to provide recommendations for a user, need to be able to use a domain-neutral user profile with preferences in one or more source domains to generate recommendations in the target domain of the proposed personalisation service.

In the following sections we will provide more details for each challenge, and describe the research questions which we can derive from each challenge.

2.2.1. Personalisation in social networking ecosystems

As discussed in section 2.1.1, a recommender system requires input data about the user and background data for the items in the recommender system inventory.

The data about user preferences and recommendable items, which is necessary in order to provide recommendations, can be collected from data sources "inside" the recommender system, or from data sources "outside" of the recommender system.

[Brusilovsky & Henze, 2007] provide a classification of recommender systems into two groups, based on their data sources:

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 corpus of background data, where all data sources are known in advance. Examples of closed corpus recommenders include the e-Commerce site Amazon, and the movie recommendation site Netflix[4].

Open corpus recommenders provide recommendations using public data sources, which are relevant for the recommendation domain and scenario. They are based on the assumption, that data sources are added at runtime. They require an open corpus, which must take discovery, selection and integration of data sources without prior knowledge into account. An example of an open corpus recommender is Google News[5], which aggregates articles from publicly accessible newspaper web sites. Users of the service can personalise their view of the aggregated news paper articles.

[4] http://netflix.com
[5] https://news.google.com/

In the classification of Brusilovsky, the crucial difference between closed corpus recommenders and open corpus recommenders, lies in the usage of public data sources for open corpus recommenders. Brusilovksy uses the term "public data sources" to refer to data sources which are available on the public web without any access control.

However, Brusilovsky's classification was a conceptual proposal. In contrast to both closed and open corpus recommenders, current social networking ecosystems are built on a different architecture which is based on different assumptions. Therefore they do not fit into the categorisation of open and closed corpus recommenders.

The architecture of social networking ecosystems is based on two central assumptions:

private sources added at run-time: Like open corpus recommender systems, social networking ecosystems can add external data sources at run-time. The data sources are provided by the third party services, which a user grants access to his user profile. However, the data provided by these external services is private, as it is controlled by the service operator of each third party service.

closed architecture: In order to give the user a sense of security, while aggregating his profile data from external services, current social networking ecosystems are built on a closed architecture. The user model and all aggregated data about a user belong to the social network operator, e.g. Facebook. This is meant to protect the privacy of the user when his profile data is exchanged between the hub site and an external third party service. However, it also creates user lock-in and data silos [Chellappa & Sin, 2005a], as the hub site is in control of all user accounts and all participating third party services.

Figure 2.1 on page 25 shows the closed architecture of the Facebook ecosystem. Facebook is the *hub site* in the center of the ecosystem. In this example, both the music personalisation service Pandora, as well as the business review and recommendation service Yelp, are third party services. Pandora and Yelp both have been granted access to the profile of the user. In its own ecosystem, Facebook prescribes the use of OpenID[6] and OAuth[7] for authentication and authorisation.

OpenID is a standard for decentralised authentication of a user. It provides a way to prove that an end user owns an identity without passing around the password of the user to a third party service. OAuth specifies a protocol for authorisation of resource access. It specifies how a user can authorise a third party service to access parts of his profile on a hub site. Instead of using a user's password to access parts of his profile at a hub site, third party services obtain access tokens which are used to access the protected resources.

As the user has authorised Pandora and Yelp to access her user profile, both services have acquired OAuth access tokens to the user profile. Whenever the user clicks the

[6] http://openid.net/specs/
[7] http://oauth.net

Facebook "Like" button on a web site, then this information is added to her user profile. This preference data in the user profile can be used by Facebook itself, and by authorised 3rd party services. In addition, Facebook provides a plugin widget for external sites that provides recommendations from that site based on the preferences of the user. In our example, the foodie site "Chow" is using the Facebook widget, to provide food recommendations for the user. Facebook uses a proprietary and secret personalisation approach to generate its recommendations. One part of this personalisation approach is the EdgeRank [Kincaid, 2010] graph algorithm. However, EdgeRank has been phased out in favour of a different secret algorithm[8].

The architecture of the Facebook ecosystem is *closed*, as Facebook is in control of the user profile. Facebook controls which users can sign-up for their ecosystem, and it also controls which 3rd party services can participate. The user can allow *external services* to access his user profile. These external services do not have to be known in advance. The data provided by the external services are *private*, their data is only available to the hub site in the ecosystem. An example of such an external service is Spotify[9].

Research question:

The closed nature of social networking ecosystems, like the ones from Facebook, Google+ and Twitter, leads to user lock-in and data silos, as the user profile data is not portable outside of the ecosystem. This leads us to the next research question:

Research question (Q2): How can we enable an open ecosystem for cross-domain recommendations outside of existing proprietary ecosystems?

What technologies should such an open ecosystem use for transporting and aggregating user profile data? Which parts of the supporting infrastructure and architecture can be decentralised, and which parts need to remain centralised?

In Chapter 6, we describe a prototype implementation, which shows how to provide the infrastructure for sharing, integrating and aggregating user profile data in an open ecosystem.

In addition, Appendix A presents a privacy-enabled architecture for open ecosystems. We describe how to combine existing infrastructure of the Web of Data and existing standards for decentralised identity management in order to achieve privacy-enabled user profile portability. The proposed architecture enables the creation of a universal private by default ecosystem with interoperability of user profile data. It complements the architecture used in the prototype of Chapter 6 with privacy features. However, implementing our proposed privacy-enabled architecture was outside of the scope of this thesis.

[8] http://marketingland.com/edgerank-is-dead-facebooks-news-feed-algorithm-now-has-close-to-100k-weight-factors-55908
[9] http://spotify.com/

2.2.2. Domain-neutral user profiles

All major social networking ecosystems cater to general purpose sharing, as they are not limited to a particular genre or domain (like music or sports) or to a specific inventory (such as books in stock at a physical warehouse). In terms of the architecture of the social networking ecosystem, this is accomplished by enabling third party services to add data about user preferences to the user profile which is stored on the hub site.

The extent to which each social network ecosystem allows sharing of user preference data varies greatly, as described on the developer pages of the respective social networks.

Facebook allows external services to share data about the user with Facebook, if the external service is authorised by the user. For instance, Spotify can add the music which a user has listened to, Yelp[10] can add restaurants which the user has rated. The data of each external service is stored in its own namespace. HTTP URIs are used to identify items. These URIs can refer to HTTP pages on Facebook or on the web site of the external service. Sharing of data between services is possible, however it requires asking the user for permission to access the namespace of another service. In addition, it requires one service to resolve the HTTP URIs of the other service, in order to identify e.g. the name of an item.

Google Plus allows external services to add data to a user profile. However, each service can only access the data which it has created itself. Sharing of user profile data between services is not possible.

Twitter allows external services to post new tweets on behalf of a user, to read the existing tweets of a user and to read the graph of followees. Each of these capabilities requires the permission of the user. Neither the creation of new preference data nor the sharing of preference data between services is allowed by Twitter.

Currently, only Facebook allows the re-use of data originating in one service by another service. However, we believe that the importance of sharing of user profile data between services will grow. It provides benefits for users, such as less lock-in to a single service and the portability of their data between personalised services [Brusilovsky & Henze, 2007]. For instance, in the future a user might be able to re-use his music preferences from the Pandora[11] music streaming service in the Spotify music streaming service.

[Abel et al., 2011] identify the re-use of preference data between services as a crucial difference between personalisation using distributed profiles, and traditional recommender systems. For the purpose of this thesis we follow [Abel et al., 2011] by distinguishing between domain-neutral user profiles and single-domain user profiles:

[10] http://yelp.com/
[11] http://www.pandora.com

single-domain user profile: only contains user preferences from one conceptual domain. Reusing of the preferences by another personalisation service is not expected.

domain-neutral user profile: user profile which is aggregated from multiple third party services. The preference data uses item identifiers which can be resolved by other personalisation services.

For current state of the art recommender systems, single-domain user profiles are mostly implemented by using vectors of item ratings or lists of plain text items [Bobadilla et al., 2013].

Vectors of item ratings are used especially by collaborative filtering implementations. Each user profile is represented by a vector which assigns a numerical rating to each recommendable item in the inventory of the system. Item rating vectors use item identifiers which are specific to the inventory of one recommender system, and as such it is very difficult to exchange item rating vectors between personalisation services with different inventories. In addition they can not represent interests outside of the inventory of a recommender service.

Lists of plain text items, can represent tags or frequent words. They are often used in content-based recommender systems. They are also called "bag-of-words". Lists of text items can represent any kind of interest independent of the inventory of a single personalisation service, and as such they enable some portability between personalisation services. However, this still leaves the problems of ambiguity and lack of synonymy [Specia & Motta, 2007] when exchanging user profiles between personalisation services.

As an alternative, graph based representations of user profiles have been emerging over the last years. As observed by [Perugini et al., 2004], the growth of social networking data has led to data structures which allow representing the graph of social connections between users. This graph of all social connections is called the *social graph*. These connections are either made directly - as a result of explicit user modeling, or indirectly - through the discovery of implicit relationships.

In addition, users of social networking ecosystems are not only directly connected to other persons, but 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 user communities follow the principle of *object centered sociality* [Bojars et al., 2008b]. The graph of all connections between users and social objects is called the *interest graph*, as it represents all interests of a user as edges in the graph.

While the concepts of the social graph and interest graph are complimentary, they both express the importance of modeling the actors in a person's social network, as well as his interests as a graph. As a result, there has been an increasing interest in using these same graphs as background data for delivering personalised services.

Research question:

Providing standards for user profile exchange inside of an ecosystem enables re-usability, portability and interoperability of meta-data and user profile data [Brusilovsky & Henze, 2007].

However, todays social networking ecosystems each use their own approaches for representing and transporting the user profile information between participating services. Google has developed the OpenSocial API [Mitchell-Wong et al., 2007] for this purpose, while Facebook has developed the Facebook Open Graph API[12].

As the OpenSocial API and the Open Graph API are incompatible with each other, no sharing of user profile data between the ecosystems of Google and Facebook is possible.

This leads us to our next research question:

Research question (Q3): What kind of data structure should be used for domain-neutral user profiles, in order to allow merging profile data from multiple personalisation services?

What kind of data model and which standards should be used for domain-neutral user profiles in our proposed open ecosystem? How can we facilitate the merging of user profiles of the same user from multiple sources?

In Chapter 6, we show how our prototype implementation uses Linked Open Data to represent domain-neutral user profiles.

2.2.3. Cross-domain recommendation

In order to leverage domain-neutral, multi-source user profiles for personalisation, a recommendation algorithm is required which can use domain-neutral user profiles with preferences in one or more domains as input data. This recommendation algorithm needs to be able to use preferences in one or more source domains in order to generate recommendations for a target domain, which is called cross-domain recommendation [Fernández-Tobías et al., 2012, Loizou, 2009, Winoto & Tang, 2008, Cremonesi et al., 2011].

In this section, we will introduce cross-domain recommendation formally and list the related research questions which we will address in this thesis.

As explained in the previous sections, social networking ecosystems enable third party personalisation services to add user preferences to the user profile which is stored on the hub site. This results in user profiles which are aggregated from multiple sources, and which contain preferences from multiple domains, depending on the contributing personalisation services.

[12] https://developers.facebook.com/docs/opengraph/

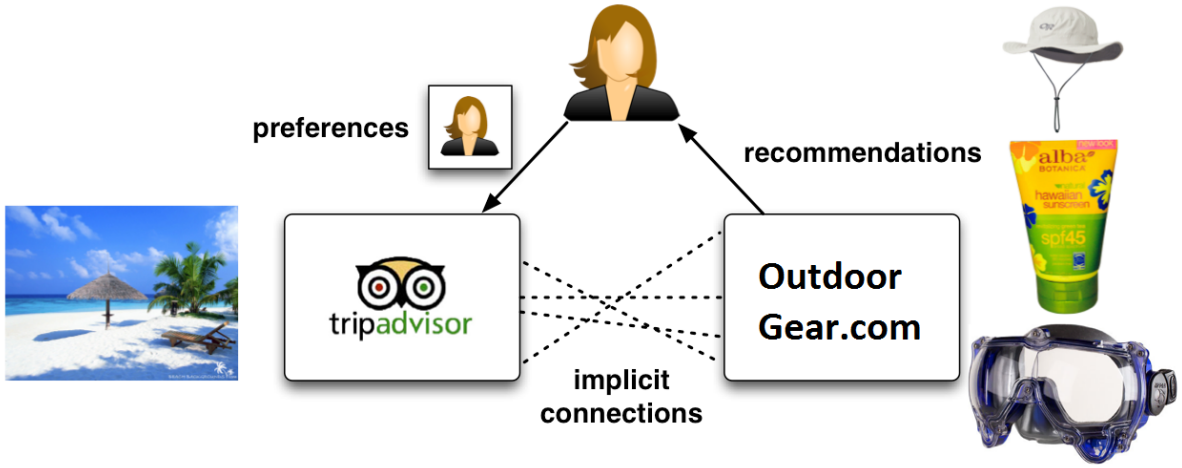

Figure 2.2.: Cross-domain personalisation use case for two federated e-commerce sites: the travel agent TripAdvisor and a hypothetical outdoor gear online retailer

Such domain-neutral, multi-source user profiles in effect enable the sharing of user preferences between personalised services. However, this introduces the possibility of the recommendation algorithm encountering a user profile without any preferences about items in the inventory of the personalised service. This can happen for instance, if a user authorises a new personalised service to access his user profile. In this situation the *new user problem* occurs, if the newly connected user profile does not contain any preferences for items in the inventory of the personalisation service. We introduced the new user problem as part of the cold-start problem in Section 2.1.3.

Figure 2.2 illustrates a situation in which a user visits a personalised site for the first time. The user just booked a trip to a tropical island via TripAdvisor, an online travel agent, and now she is shopping for travel accessories at an outdoor gear online retailer, which she never has used before. The online retailer can not provide recommendations using single-domain personalisation, because the outdoor retailer does not know any preferences of the new user. In this example, the *source domain* of the recommendation are the user preferences in the travel domain. The *target domain* of the recommendation are the different items of outdoor gear in the site inventory of the online retailer.

However, if TripAdvisor and OutdoorGear.com are both participants in the same personalised ecosystem, then the user can allow OutdoorGear.com to access her user profile. This will allow sharing and re-using of user profile data between the two personalised services. In this way, OutdoorGear.com could use a *cross-domain recommendation* algorithm, to identify connections between the travel items in its site inventory and the travel interests which TripAdvisor knows about the user. In the example, OutdoorGear.com could recommend a sun hat, scuba googles, sun screen or a guide book for the tropical island.

Definition of Domain

Different other approaches exist for formalising both the notion of a domain as well as the recommendation task in relation to cross-domain recommendation:

Regarding the *notion of a domain*, [Loizou, 2009] specify that a domain is characterised by a group of experts who have expressed preferences on a set of items. In a similar way, [Cremonesi et al., 2011] defines a domain A as having a set of users U_A, a set of items I_A and a user-rating matrix R_A. However, this definition already assumes that the recommendation algorithm uses ratings, so we will not include the rating matrix R_A in our notion of a domain. Instead we will use the more general notion of a preference to connect the users and items of a domain.

We define the terms *domain*, *source domain* and *target domain* as follows:

Domain: A domain D has an associated set of items I_D, and an associated set of users U_D. The users have preferences about the items in the domain, which we express as $pref(u \in U_D) = \{p_{u1}, \ldots, p_{un}\}$, where $n = |I_D|$ is the number of items in I_D. p_{un} is the preference of user u for item $i_n \in I_D$. We leave the specific details of the preferences p_{un} open, as they are dependent on the recommendation algorithm.

Source Domain: A source domain D_S is a domain for which a recommender system has user preference data available. We express this as requiring non-empty user preferences for all users in the set of users U_{D_S} of D_S, so that $pref(u) \neq \emptyset \; \forall u \in U_{D_S}$.

Target Domain: A domain D_T which contains a subset of the items of an organisation for which the recommender system is used.

Please note, that our definition of a domain is independent of the type of items which belong to a domain. E.g. a domain does not have to be defined through item types such as “book”, “music” or “travel”. Instead, our domain definition uses the boundaries of the organisations in which the cross-domain recommender system is used.

In addition, please note that our definition of domain, source domain and target domain does not account for changing the contents of the domains over time. Each instance of a domain, source domain or target domain is a *snapshot* of the contents of the specific domain at a specific time. Changes to the domains are possible, e.g. new users can be add to a source domain. In this case, the new domain which includes the new user has to be used instead of the old one.

Examples of domains:

- For the example in Figure 2.2, the domain of TripAdvisor $D_{TripAdvisor}$ is the source domain, and the domain of OutdoorGear.com $D_{OutdoorGear.com}$ is the target domain.

- $D_{AmazonBooks}$ is the source domain, with the set of all the books $I_{AmazonBooks}$ on Amazon.com, and the users $U_{AmazonBooks}$ of Amazon which rated those books. The preferences are the star ratings of the user.
- $D_{ITRugby}$ is the source domain, which has the set of all news articles about the sport Rugby on the Irish Times web site as $I_{ITRugby}$. The set of users $U_{ITRugby}$ then is the set of users which have commented on these news articles, as the Irish Times does not allow readers to rate articles. In this case, the preferences are the binary connections between users and the articles on which they commented.

The cross-domain recommendation task

Regarding the definition of the *cross-domain recommendation task*, we first need to specify the amount of overlap between the target and source domains. [Cremonesi et al., 2011] differentiates between four situations in which cross-domain personalisation is applicable:

- No overlap: $U_S \cap U_T = \emptyset \wedge I_S \cap I_T = \emptyset$
- User overlap: $U_S \cap U_T \neq \emptyset$
- Item overlap: $I_S \cap I_T \neq \emptyset$
- Full overlap: $U_S \cap U_T \neq \emptyset \wedge I_S \cap I_T \neq \emptyset$

Of these four situations, we only consider the *no overlap* situation as requiring a cross-domain recommendation algorithm. For the other three situations, the preference data of the two domains can be merged, after which a regular single-domain recommendation algorithm can be used on the data [Cremonesi et al., 2011].

Definition of the cross-domain recommendation task: We define the cross-domain recommendation task as follows: Let $D_S = (U_S, I_S)$ be the source domain, and let $D_T = (U_T, I_T)$ be the target domain. Further, we require that there is no overlap between the source and target domains, so that $U_S \cap U_T = \emptyset \wedge I_S \cap I_T = \emptyset$. Then the task of performing a cross-domain recommendation $rec(u \in U_S) = \{i_1, \ldots, i_r | i_j \in I_T\}$ for user $u \in U_S$, consists of recommending a set of items I_T of the target domain D_T.

For the definition of the cross-domain recommendation task itself, [Fernández-Tobías et al., 2012] propose two weaker definitions of the cross-domain recommendation task, which requires an overlap between the target and source domain. The first task, is to exploit data in the source domain for improving the quality of the recommendations for items in the target domain. This assumes item overlap. The second task, is to make joint recommendations for items belonging to different domains, so that items in $I_S \cup I_T$ are recommended to users in $U_S \cup U_T$. However, this still requires some overlap between target and source domain. We do not consider these two weaker definitions, as any overlap between the

target and source domain allows using a regular single-domain recommendation algorithm [Cremonesi et al., 2011].

Requirements for our proposed cross-domain recommendation algorithm

Given our definition of the cross-domain recommendation task, a recommendation algorithm is required which can perform this task as part of our cross-domain personalisation framework.

Further, we want to ensure that recommendations can be provided even if the available rating data for the source domain is very sparse. The worst case scenario regarding rating sparsity is to have no ratings at all for the target domain for which recommendations have to be made. Therefore, we make it a requirement for our cross-domain recommendation algorithm to not use any rating data from the target domain at all.

We can summarise the requirements for the recommendation algorithm which we will use in our cross-domain framework as follows:

1. The algorithm must be able to perform the cross-domain recommendation task without any overlap between the source and target domain.
2. The algorithm must not require any rating data for the target domain.

Research question:

Facebook provides a so-called "social plugin" for cross-domain recommendations, which allows an external site to recommend content or items from its inventory to users without prior interest in the domain of the external site. E.g. a weblog about rugby, can use this plugin to recommend its articles to users without prior preferences about "sports". However, Facebook's cross-domain recommendations use *a proprietary algorithm*, so that users without a Facebook account can not benefit from these recommendations.

In order to provide cross-domain recommendations in our open framework, a non-proprietary algorithm is required. Both [Fernández-Tobías et al., 2012] and [Cremonesi et al., 2011] agree that collaborative filtering can not be used when there is no overlap between domains. Fernandez-Tobias et al. suggest using a hybrid recommender system with a content-based component without providing more details on the algorithm.

Following along similar lines, [Cremonesi et al., 2011] also present an approach which uses a graph-based algorithm to add inter-domain connections which then can be exploited by collaborative filtering. However, the graph-based algorithm is not used to generate the recommendations, it is only used to add additional connections between domains. As collaborative filtering is used to generate the recommendations, the proposed approach does require ratings in the target domain.

None of the algorithms from the related work are able to fulfill both requirements for the recommendation algorithm in our cross-domain framework.

The related work in the area of cross-domain recommendation suggests that a graph-based approach, which is able to find indirect connections between domains is required, in order to perform cross-domain recommendations when there is no overlap between domains. However, the question of which specific algorithm to use remains open.

Research question (Q4): Which recommendation algorithm can generate cross-domain recommendations, without any overlap between source and target domain and without any ratings for the target domain?

In Chapter 4, we will introduce our algorithm for cross-domain personalisation. It fulfills both requirements of our framework, as it does not require an overlap between source and target domain, and it does not require any ratings in the target domain.

2.3. Using Linked Data for Cross-Domain Recommendation

As we have shown in the previous section, data with connections between different domains is required in order to perform cross-domain recommendations. In this section, we introduce Linked Data as an enabling technology for cross-domain recommendation. As we will show, Linked Data can provide the indirect links between different domains.

The Linked Data principles, as well as Semantic Web technologies such as the Resource Description Format (RDF), prescribe a standardised, open and interoperable way for expressing graph data. The Linked Data principles encourage data publishers to publish linkage data about connections between different data sets. In addition, Linked Data is available about many different conceptual domains. This makes Linked Data an interesting source for concept identifiers from different domains, as well as cross-domain linkage data.

In this section we first describe how Linked Data and the Web of Data have emerged. Then we describe the graph based data model of RDF and the Linked Data principles. Then we list the most prominent sources of Linked Data from social web sites, broadcasters and news publishers, scientific publications, e-commerce sites and public government data.

However, while Linked Data has the potential to provide the background data required for cross-domain recommendations, many guidelines and patterns for publishing, accessing and consuming Linked Data are currently missing. We will list some of the most prominent cases of missing implementation guidelines for Semantic Web technologies. This in turn

introduces the research question of how to implement the functionality on an application level, which is required in order to leverage Linked Data and Semantic Web technologies.

2.3.1. The emergence of the Web of Data

Starting in the early 2000s, the World Wide Web consortium has been developing a new set of technologies for the Web, together with the scientific community. The goal of these new technologies was to simplify knowledge-intensive applications, by enabling a Web of interoperable and machine-readable data [Berners-Lee et al., 2001] based on formal and explicit descriptions of the structure and semantics of the data [Decker et al., 2000]. The umbrella term for this new set of technologies is *Semantic Web*. The benefits of Semantic Web technologies include simplification of information retrieval [Abecker & van Elst, 2004], information extraction [Davies et al., 2002] and data integration [Fensel et al., 2001].

Semantic Web technologies and standards had a limited uptake up 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[13], which makes free and public data available as Linked Data. Figure 2.3 shows the sources in the Linking Open Data cloud as of September 2011. At the time of writing, approximately 300 different sources following the Linked Data principles are available.

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* [Bizer et al., 2009] 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.

2.3.2. The Linked Data Principles

The Web of Data is an 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.

[13] http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData/

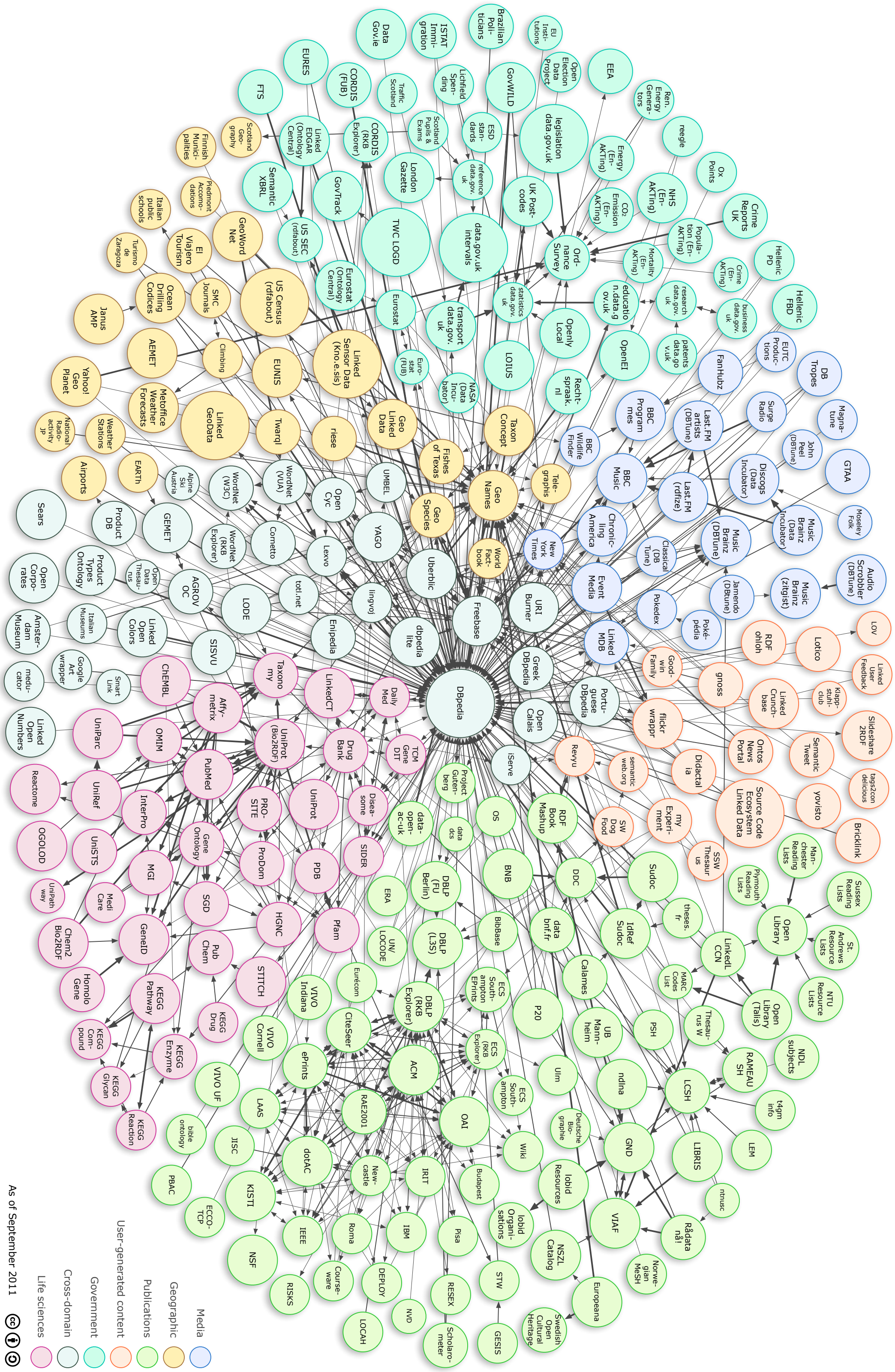

Figure 2.3.: Sources in the Linking Open Data cloud as of September 2011.

The Web of Data utilises a subset of technologies from the Semantic Web technology stack: the *Resource Description Framework (RDF)* [Decker et al., 2000] 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* [Bizer et al., 2009], 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 particular, the fourth principle encourages data publishers to include links from their data source, to other data sources. These links then provide inter-domain linkage, which can provide the links between domains, which are necessary for cross-domain recommendations.

2.3.3. Available Linked Data sources

As Figure 2.3 on page 38 shows, currently there are already Linked Data sources from many different areas 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**[14] [Lehmann et al., 2014], which extracts RDF from Wikipedia topic pages, and thus provides URIs and RDF data about topics from a large number of areas and topics. 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 `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`.

Broadcasters and news publishers provide data about media content and usage. As an example, the DBTune project [Raimond et al., 2008] 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 source of Linked Data, such as DBpedia or Musicbrainz[15]. 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 area of life sciences, due to the knowledge intensive nature of much research in this area. Some of these sources include UniProt, PubMed and the Gene Ontology.

E-commerce sites can use the Good Relations vocabulary[16] 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[17], the US government provides their data[18] in different legacy formats such as spreadsheets and text files. Parts of the US data are then converted to Linked Data by the community[19].

[14] http://dbpedia.org
[15] http://musicbrainz.org/
[16] http://www.heppnetz.de/projects/goodrelations/
[17] http://data.gov.uk/
[18] http://www.data.gov/
[19] http://data-gov.tw.rpi.edu/

2.3.4. Missing implementation guidelines for Semantic Web technologies

Most Semantic Web technologies are standardised without providing explicit guidelines for the implementation of the standards. This can lead to incompatible implementations especially when the interplay of multiple standards is involved. The establishment of a community consensus generally requires comparison and alignment of existing implementations. However, the waiting period until agreed guidelines and patterns for the standard have been published can have a negative impact on the adoption of a standard. To illustrate this research challenge, we discuss the most visible instances of missing guidelines and standards. These are related to (1) publishing and consuming data, (2) embedding of RDF on web pages, (3) writing data to a remote store, and (4) restricting read and write access.

Publishing and consuming data

All Linked Data applications consume RDF data of some form. However there are several incompatible possible implementations for publishing of RDF data. As we will show through a survey of RDF-based applications in Chapter 3, only after the publication of the Linked Data principles [Berners-Lee, 2006] in 2006 and the best practices for publishing Linked Data [Bizer et al., 2007] in 2007, did an interoperable way for publishing of data emerge.

Embedding RDF on web pages

While the majority of Linked Data applications have web pages as user interfaces, RDF data usually is published or stored separately from human readable content. This results in out-of-sync data, incompatibilities and difficulties for automatically discovering data. After publishing the first version of the RDFa standard[20] in 2008, embedding of RDF on web pages strongly increased. The 1.1 version of RDFa has been a W3C recommendation since August 2013.

Writing data to a remote store

While SPARQL standardised remote querying of RDF stores, it did not include capabilities for updating data. Together with a lack of other standards for writing or updating data, this results in a lot of applications which do not include a user interface for authoring data.

[20] http://www.w3.org/TR/rdfa-primer/

The lack of capabilities of creating, editing or deleting data is addressed by SPARQL Version 1.1[21].

Restricting read and write access

One capability that is currently mostly missing from all Semantic Web standards is the management of permissions for read or write access to resources and RDF stores. If such capabilities are required for an application, then they will be implemented on a case-by-case basis without the use of any related standards.

2.3.5. Research Question

As we have shown in this section, the Linked Data principles and the resulting Linking Open Data (LOD) cloud, have the potential to provide both concept identifiers from many different domains, as well as cross-domain linkage data. In particular, the fourth Linked Data principle encourages data publishers to include links from their data source, to other data sources. These links then provide inter-domain linkage, which can provide the links between domains, which are necessary for cross-domain recommendations.

However, the guidelines for implementing the functionality which is required for leveraging Semantic Web technologies are not clearly and completely defined, as we have discussed. In consequence, a blue print for implementing applications which use Linked Data is required.

In terms of software architecture, such a blue print for individual applications, is provided by a conceptual architecture. As defined by Soni et al. [Soni et al., 1995], a *conceptual architecture* describes a system in terms of its major design elements and the relationships among them, using design elements and relationships specific to the domain. It enables the discussion of the common aspects of implementations from a particular domain, and can be used as a high-level architectural guideline when implementing a single application instance for that domain.

This leads us to the following research question:

Research question (Q5): Which components and best practices need to be implemented by a recommender system in order to leverage Linked Data for recommendations ?

In Chapter 3, we derive a conceptual architecture for Linked Data applications from an empirical survey of RDF-based applications. This conceptual architecture describes different areas of functionality, which a recommender system can then implement in order to leverage Linked Data.

[21] http://www.w3.org/TR/sparql11-query/

2.4. Summary

In this chapter, we have introduced the background, which motivates the research presented in this thesis. In particular, we have described the limitations of current state-of-the art recommendation approaches regarding the cold-start problem. Then we introduced the requirements for our cross-domain personalisation approach, which are derived from the requirements of personalisation in current social networking ecosystems. In the last part of this chapter, we introduced Linked Open Data as an enabling technology for our cross-domain personalisation framework.

We have introduced the research questions for our thesis in more detail:

Research question (Q1): How can we mitigate the cold-start problem in order to provide recommendations for new users?

Research question (Q2): How can we enable an open ecosystem for cross-domain recommendations outside of existing proprietary ecosystems?

Research question (Q3): What kind of data structure should be used for domain-neutral user profiles, in order to allow merging profile data from multiple personalisation services?

Research question (Q4): Which recommendation algorithm can generate cross-domain recommendations, without any overlap between source and target domain and without any ratings for the target domain?

Research question (Q5): Which components and best practices need to be implemented by a recommender system in order to leverage Linked Data for recommendations?

Current recommendation algorithms can be grouped in five classes: Collaborative filtering, demographic filtering, content-based filtering, knowledge-based filtering as a special case of content-based filtering and hybrid filtering approaches. The cold-start problem can appear in three cases, for (a) new items, (b) new users and when (c) the background data of the recommender system is very sparse. In order to mitigate the cold start problem, a common strategy is to turn to additional information sources. In order to achieve this, CF algorithms are often combined with content-based or knowledge-based recommendation in a hybrid. However, while using a Collaborative Filtering algorithm in a hybrid recommender system can mitigate the challenge of sparse data and new items, hybrid approaches are not able to address the problem of a new user without any known preferences.

The new user problem is important for our cross-domain personalisation framework, as our proposed conceptual architecture for personalised services allows the sharing of user profiles. In this situation the new user problem occurs, if the newly connected user profile does not contain any preferences for items in the inventory of the personalisation

service. Our cross-domain personalisation framework provides an approach to mitigate the cold-start problem for new users.

In the second part of this chapter, we derived the requirements for our cross-domain personalisation approach from the requirements of personalisation in current social networking ecosystems.

Social networking ecosystems introduce new requirements for personalisation:

Multi-source user profiles: Each user of a hub site has a user profile, with parts originating from multiple third party services. Each ecosystem must provide an architecture and a transport mechanism for aggregating such a multi-source user profile from multiple third party services.

Domain-neutral user preferences: The user preferences of one user will not just be about items from one domain, such as movies. Instead the preferences will span the domains of multiple services that have contributed to the user profile, thus resulting in the user profile being domain-neutral.

Cross-domain recommendations: Third party services which want to provide recommendations for a new user, need to be able to use preferences outside of their own domain in order to recommend items from their own inventory. This is enabled by a cross-domain recommendation algorithm that can use preferences in a source domain to recommend items in a different source domain.

We defined a domain as any set of recommendable items, together with a set of users with preferences about the recommendable items. Our formal definition of a domain is in Section 2.2.3.

We defined the cross-domain recommendation task, as providing recommendations from a target domain using preferences in a different source domain, without any overlap between the source and target domain.

Based on our discussion of the cross-domain recommendation task and the relevance of the cold-start problem for cross-domain personalisation, we listed the requirements for our cross-domain algorithm, as follows:

1. The algorithm must be able to perform the cross-domain recommendation task without any overlap between the source and target domain.
2. The algorithm must not require any rating data for the target domain.

In the last part of this chapter, we introduced Linked Open Data as an enabling technology for our cross-domain personalisation framework.

In order to provide cross-domain recommendations, a source of background data with connections between items from the target and source domains is required. Linked

Open Data (LOD) is available about many different conceptual domains. LOD provides concept identifiers from many different domains, as well as cross-domain linkage data. However, the are no guidelines or best practices for implementing the functionality which is required for leveraging Linked Open Data for recommender systems available.

In this thesis, we introduce an open framework for cross-domain personalisation using semantic interest graphs from Linked Open Data. It uses Linked Open Data in order to enable cross-domain recommendations. The framework consists of a conceptual architecture for Linked Open Data recommender systems and a cross-domain recommendation algorithm. The conceptual architecture for LOD recommender systems provides guidelines and best practices on how to share, aggregate and integrate user profile data using Linked Open Data. The cross-domain recommendation algorithm uses inter-domain linkage data from Linked Open Data in order to provide cross-domain recommendations. It fulfills our requirements for a suitable cross-domain recommendation algorithm, as it does not require an overlap between source and target domain, and because it does not require any rating data for the target domain.

Chapter 3.

Conceptual Architecture for Linked Open Data Recommender Systems*

Personalisation in open ecosystems, where user profile data can be shared between the personalised systems, requires an infrastructure which can support aggregation and integration of preference data from multiple sources, as well as an approach to represent user preferences that is domain-neutral. Enabling multi-source and domain-neutral user preferences then in turn can be leveraged by a cross-domain recommendation algorithm.

We propose to use Semantic Web technologies and Linked Open Data (LOD) to provide the supporting infrastructure for personalisation in open ecosystems. Semantic Web technologies facilitate data aggregation and integration. Linked Open Data provides concept identifiers from many different domains as well as cross-domain linkage data, which provides indirect connections between concepts from different domains.

However, there are no established best practices and guidelines for implementing the functionality which is required to leverage Linked Open Data (LOD). Therefore the goal of this chapter, is to review the best practices for implementing LOD-based applications, with an additional focus on the functionality required for recommender systems. Towards this goal, we present a conceptual architecture for recommender systems using Linked Open Data. The proposed conceptual architecture is derived from an empirical survey of RDF-based applications, and describes the components and their high-level functionality, which a recommender system can implement in order to leverage Linked Open Data.

*This chapter is based on previously published work:

1. Benjamin Heitmann, Richard Cyganiak, Conor Hayes, Stefan Decker, "Architecture of Linked Data Applications", In Andreas Harth, Katja Hose, Ralf Schenkel (ed.), *Linked Data Management: Principles and Techniques in the Series on Emerging Directions in Database Systems and Applications*, CRC Press, United States, 2014.

2. Benjamin Heitmann, Richard Cyganiak, Conor Hayes, Stefan Decker, "An Empirically-Grounded Conceptual Architecture for Applications on the Web of Data", in *IEEE Transactions on Systems, Man and Cybernetics, Part C – Applications and Reviews; Special Issue on Semantics Enabled Software Engineering*, 2012.

Contributions of this chapter: We address research question (Q5) in this chapter. (Q5) asks for the best practices for leveraging Linked Open Data for recommender systems. We present a conceptual architecture for Linked Open Data recommender systems that provides a blueprint for consuming, integrating and publishing Linked Open Data. We instantiate this architecture in Chapter 6, where we describe our prototype implementation.

Chapter outline: First, in Section 3.1, we discuss the requirements of recommender systems in open ecosystems, and how the requirements relate to Linked Open Data. Then, in Section 3.2, we present the results of an *empirical survey* which we have performed on RDF-based applications over most of the past decade, from 2003 to 2009. As the Linked Data principles where introduced in 2006, this allows us to describe the state-of-the-art for developing and deploying applications which use RDF and Linked Data.

We then use the results of this survey in Section 3.3, as the empirical foundation for a component-based *conceptual architecture* for Linked Data applications in general. Our conceptual architecture describes the high level components which are most often observed among the surveyed applications. These components implement the functionality that differentiates applications using RDF from database-driven applications. For each component we describe the most common implementation strategies that are suited to specific application goals and deployment scenarios.

Then, in Section 3.4, we describe how to customise the proposed generic conceptual architecture for the special case of recommender systems using Linked Open Data.

In addition, we present the results of the analysis of an example application in Section 3.3.4, and we list related work in the areas of empirical surveys about the adoption of Semantic Web technologies and architectures for Semantic Web applications in Section 3.5.

3.1. Background: Using Linked Open Data for Open Recommender Systems

As identified in the background Chapter (Chapter 2), the three main requirements for recommender systems in open, personalised ecosystems are as follows:

Multi-source user profiles: Aggregating and integrating of user profiles from multiple sources is required, as the preferences of a single user can be distributed across multiple systems in the ecosystem.

Domain-neutral user profiles: Representing and merging of user preferences from multiple domains is required, as each source of user preferences, that contributed to a multi-source user profile, can describe a different domain.

Cross-domain personalisation: A personalisation approach is required which can leverage user preferences in a source domain to provide recommendations from a different target domain.

We propose to use Semantic Web technologies and Linked Open Data (LOD) to provide the supporting infrastructure for personalisation in open ecosystems. Semantic Web technologies [Decker et al., 2000] facilitate data aggregation and integration. Linked Open Data [Berners-Lee, 2006] can provide inter-domain linkage data for items from two different domains.

As we discussed in the Background Section, Linked Open Data is an enabling technology for cross-domain linkage data between items from different source and target domains for three reasons:

1. The Linked Data principles and the graph data model of RDF specify a standardised, open and interoperable way for *expressing graph data*.
2. The Linking Open Data cloud provides data sets from many different conceptual areas which have been published by many different data providers using RDF and the Linked Data principles. These data sets provide *re-usable concept identifiers*, in the form of RDF URIs.
3. In addition, the Linking Open Data cloud provides *cross-domain links between concepts from many different conceptual areas*, as the Linked Data principles explicitly encourage data providers to publish links between concepts from their Linked Data sets and concepts from other Linked Data sets.

However, the guidelines and best practices for implementing the functionality which is required for leveraging RDF-based data and Semantic Web technologies are not clearly and completely defined.

Such guidelines and best practices, are required in order to help developers to move to an RDF-based data representation. Such a move to an RDF-based data representation introduces challenges for the application developer in rethinking the Web application outside the standards and processes of database-driven Web development. These include, but are not limited to, the graph-based data model of RDF [Decker et al., 2000], the Linked Data principles [Berners-Lee, 2006], and formal and application specific semantics [Bizer, 2009].

This in turn introduces the research question of how to leverage Linked Data for a recommender system. We can generalise this research question for all software applications, by aiming to identify the functionality which is required by a software application in order to leverage RDF, Linked Data and Semantic Web technologies. In consequence, a blue print for implementing applications which use Linked Data is required. In terms of software architecture, such a blue print for individual applications, is provided by a conceptual architecture [Soni et al., 1995].

In order to derive such a blue print for recommender systems using Linked Open Data in this Chapter, we first perform an empirical survey of RDF-based applications. We use the results of the survey as the empirical grounding for proposing a conceptual architecture for generic Linked Data applications. We then customise the proposed architecture for Linked Open Data recommender systems.

3.2. An empirical survey of RDF-based applications

The evolution of Linked Data and Semantic Web technologies is characterised by the constant introduction of new ideas, standards and technologies, and thus appears to be in a permanent state of flux. However, more then a decade has passed since Tim Berners-Lee et al. published their comprehensive vision for a Semantic Web [Berners-Lee et al., 2001] in 2001. This allows us to look back on the empirical evidence from RDF-based applications that have been developed and deployed during this time.

[Sjoberg et al., 2007] argue that there are relatively few empirical studies in software engineering research, and that the priority for evaluating new technologies is lower than that of developing new technologies. We can transfer this observation to research on the engineering of RDF-based and Linked Data applications. Without empirical evidence it is difficult to evaluate the adoption rate of Linked Data and Semantic Web technologies. To provide such insights, we collect evidence from a representative sample of the population of developers deploying applications using RDF or Linked Data. In particular we perform an empirical survey of over a hundred applications from two demonstration challenges that received much attention from the research community.

The results of our survey enable us to determine the state of adoption of key research goals such as data reuse, data integration and reasoning. In this section, we first explain our research method. Then the findings of our empirical survey are presented, after which we discuss threats to the validity of the survey.

3.2.1. Research method of the survey

The empirical evidence for our survey is provided by 124 applications that were submitted to two key demonstration challenges in the Semantic Web research community:

1. the "Semantic Web challenge"[1] [Klein & Visser, 2004] in the years 2003 to 2009 with 101 applications, which is organised as part of the International Semantic Web Conference.

[1] http://challenge.semanticweb.org/submissions.html

2. the "Scripting for the Semantic Web challenge"[2] in the years 2006 to 2009 with 23 applications, which is organised as part of the European Semantic Web Conference.

As the Semantic Web was an emerging research topic, the majority of applicants were from research centres or university departments, though there were some industrial participants.

Each challenge awards prizes for the top entries, as selected by the judges of the respective contest. In order to apply, the developers of an application were required to provide the judges with access to a working instance of their application. In addition, a scientific paper describing the goals and the implementation of the application was also required.

The applications which were submitted to the "Semantic Web challenge" had to fulfill a set of minimal requirements, as described in [Klein & Visser, 2004]: The information sources of the application had to be geographically distributed, and should have had diverse ownership so that there was no control of the evolution of the data; the data was required to be heterogeneous and from a real-world use-case. The submitted applications should have assumed an "open world model", meaning that the information never was complete. Lastly the submitted applications were required to use a formal description of the data's meaning. Applications submitted to the "Scripting for the Semantic Web challenge" only had to be implemented using a scripting programming language.

The applications from the two challenges are from very different application areas, such as life sciences, cultural heritage, geo-information systems, as well as media monitoring, remote collaboration and general application development support. Example applications include, the Bio2RDF [Nolin et al., 2008] project, which has converted 30 life sciences data-sets into RDF and then generated links between the data sets in a consistent way. DBpedia Mobile [Becker & Bizer, 2008] is a location-aware client for mobile phones which displays nearby locations taken from DBpedia. As a related project, LinkedGeoData [Auer et al., 2009] transforms data from the OpenStreetMap project into RDF and interlinks it with other spatial data sets. Then there is the Semantic MediaWiki [Krötzsch et al., 2006] platform, which extends the MediaWiki platform used by Wikipedia with semantic capabilities, such as templates for entities and consistency checks. On the more applied side of the spectrum, there is MediaWatch [Scharl et al., 2007], which crawls news articles for concepts related to climate change, extracts ontology concepts and provides a unified navigation interface. The MultimediaN E-Culture demonstrator [Schreiber et al., 2006] enables searching and browsing of cultural heritage catalogs of multiple museums in the Netherlands. As a final example, NASA has developed the SemanticOrganiser [Keller et al., 2004] which supports the remote collaboration of distributed and multidisciplinary teams of scientists, engineers and accident investigators with Semantic technologies.

[2] http://www.semanticscripting.org

We collected the data about each application through a questionnaire that covered the details about the way in which each application implemented Semantic Web technologies and standards. It consisted of 12 questions which covered the following areas:

1. Usage of programming languages, RDF libraries, Semantic Web standards, schemas, vocabularies and ontologies.
2. Data reuse capabilities for data import, export or authoring.
3. Implementation of data integration and schema alignment.
4. Use of inferencing.
5. Data access capabilities for the usage of decentralised sources, usage of data with multiple owners, data with heterogeneous formats, data updates and adherence to the Linked Data principles.

The full data of the survey and a description of all the questions and possible answers is available online[3].

For each application the data was collected in two steps: First, the application details were filled into the questionnaire based on our own analysis of the paper submitted with the application. Then we contacted the authors of each paper, and asked them to verify or correct the data about their application. This allowed us to fill in questions about aspects of an application that might not have been covered by a paper. For instance the implementation details of the data integration are not discussed by most papers. 65% of authors replied to this request for validation.

3.2.2. Findings

The results of our empirical survey show that the adoption of the capabilities that characterise Semantic Web technologies has steadily increased over the course of the demonstration challenges. In this section we present the results of the survey and analyse the most salient trends in the data. Tables 3.1, 3.2 and 3.3 summarise the main findings.

Table 3.1 shows the survey results about the programming languages and RDF libraries that were used to implement the surveyed applications. In addition, it shows the most implemented Semantic Web standards and the most supported schemas, vocabularies and ontologies. Java was the most popular programming language choice throughout the survey time-span. This accords with the fact that the two most mature and popular RDF libraries, Jena and Sesame both require Java. On the side of the scripting languages, the most mature RDF library is ARC, which explains the popularity of PHP for scripting Semantic Web applications. From 2006 there is a noticeable consolidation trend towards supporting RDF, OWL and SPARQL, reflecting an emergent community consensus. The

[3] http://the-blank.net/semwebappsurvey/

Table 3.1.: Implementation details by year, top 3 entries per cell

	2003	2004	2005	2006
Programming languages	Java 60% C 20%	Java 56% JS 12%	Java 66%	Java 10% JS 15% PHP 26%
RDF libraries	—	Jena 18% Sesame 12% Lucene 18%	—	RAP 15% RDFLib 10%
SemWeb standards	RDF 100% OWL 30%	RDF 87% RDFS 37% OWL 37%	RDF 66% OWL 66% RDFS 50%	RDF 89% OWL 42% SPARQL 15%
Schemas/ vocabularies/ ontologies	RSS 20% FOAF 20% DC 20%	DC 12% SWRC 12%	—	FOAF 26% RSS 15% Bibtex 10%

	2007	2008	2009	overall
Programming languages	Java 50% PHP 25%	Java 43% PHP 21%	Java 46% JS 23% PHP 23%	Java 48% PHP 19% JS 13%
RDF libraries	Sesame 33% Jena 8%	Sesame 17% ARC 17% Jena 13%	Sesame 23%	Sesame 19% Jena 9%
SemWeb standards	RDF 100% SPARQL 50% OWL 41%	RDF 100% SPARQL 17% OWL 10%	RDF 100% SPARQL 69% OWL 46%	RDF 96% OWL 43% SPARQL 41%
Schemas/ vocabularies/ ontologies	FOAF 41% DC 20% SIOC 20%	FOAF 30% DC 21% DBpedia 13%	FOAF 34% DC 15% SKOS 15%	FOAF 27% DC 13% SIOC 7%

survey data about vocabulary support requires a different interpretation: While in 2009 the support for the top three vocabularies went down, the total number of supported vocabularies went from 9 in 2003 to 19 in 2009. This can be explained by the diversity of application domains for which Linked Data became available.

The simplification of data integration is claimed as one of the central benefits of implementing Semantic Web technologies. Table 3.2 shows the implementation of data integration grouped by implementation strategy and year. The results suggest that, for a majority of the surveyed applications, data integration still requires manual inspection of

Table 3.2.: Data integration by implementation strategy and year

	2003	2004	2005	2006	2007	2008	2009
manual	30%	13%	0%	16%	9%	5%	4%
semi-automatic	70%	31%	100%	47%	58%	65%	61%
automatic	0%	25%	0%	11%	13%	4%	19%
not needed	0%	31%	0%	26%	20%	26%	16%

the data and human creation of rules, scripts and other means of integration. However the number of surveyed applications that implement fully automatic integration is on the rise, while the number of surveyed applications which require manual integration of the data is steadily declining. The steady number of surveyed applications that do not require data integration, can be explained by the availability of homogeneous sources of RDF and Linked Data that do not need to be integrated.

Table 3.3 shows the survey results on the support for particular features over the time period. Some capabilities have been steadily supported by a majority of surveyed applications throughout the whole period, while other capabilities have seen increases recently or have suffered decreases.

Support for decentralised sources, data from multiple owners, and sources with heterogeneous formats is supported each year by a large majority of the surveyed applications. We can attribute this to the central role that these capabilities have in all of the early foundational standards of the Semantic Web such as RDF and OWL. While support for importing and exporting data has fluctuated over the years, it still remains a popular feature. Since 2007, the SPARQL standard has been increasingly used to implement APIs for importing data.

The Linked Data principles are supported by more then half of the contest entries in 2009, after being formulated in 2006 by Tim Berners-Lee [Berners-Lee, 2006]. Interestingly the increasing usage of the Linking Open Data (LOD) cloud may explain some of the positive and negative trends we observe in other capabilities. For example, the negative correlation (-0.61) between support for linked data principles and inferencing is probably explained by the fact that data from the LOD cloud already uses RDF and usually does not require any integration.

Support for the creation of new structured data is strongly correlated (0.75) with support for the Linked Data principles. This can be explained by the growing maturity of RDF stores and their APIs for creating data, as well as increased community support for adding to the Linked Data cloud. On the other hand, support for updating data after it

Table 3.3.: Results of the binary properties from the application questionnaire

	2003	2004	2005	2006	2007	2008	2009
Data creation	20%	37%	50%	52%	37%	52%	76%
Data import	70%	50%	83%	52%	70%	86%	73%
Data export	70%	56%	83%	68%	79%	86%	73%
Inferencing	60%	68%	83%	57%	79%	52%	42%
Decentralised sources	90%	75%	100%	57%	41%	95%	96%
Multiple owners	90%	93%	100%	89%	83%	91%	88%
Heterogeneous formats	90%	87%	100%	89%	87%	78%	88%
Data updates	90%	75%	83%	78%	45%	73%	50%
Linked Data principles	0%	0%	0%	5%	25%	26%	65%

was acquired is strongly negatively correlated (-0.74) with the growth in support for the LOD principles. This could reflect a tendency to treat structured data as being static.

3.2.3. Discussion of threats to validity

We will discuss and deflect the three most commonly raised threats to the validity of our empirical survey. These are (i) the representativeness, (ii) the number of authors validating the analysis of their applications and (iii) the reliance on analysing academic papers instead of source code.

The first threat to validity concerns the selection of applications for our empirical survey. We choose to use only applications from two academic demonstration challenges for our survey. This may raise the question of representativeness, as most of the applications are academic prototypes. However, as both challenges explicitly also invited and received some industry submissions, the two challenges do not have a purely academic audience.

In addition, several of the applications from the "Semantic Web challenge" were successfully commercialised in the form of start-ups: The Sindice [Oren & Tummarello, 2007] crawler and lookup index for Linked Data resources became the foundation for SindiceTech.com which provides products in the area of knowledge data clouds. The Seevl application [Passant, 2011], a challenge winner in the year 2011, provides music recommendations

and additional context for musical artists on YouTube and became a successful start-up. The Event Media [Khrouf et al., 2012] application aggregates information about media events, and interlinks them to Linked Data. Event Media was a winner of the challenge in 2012, and entered a partnership with Nokia.

Furthermore, we believe that the data of our survey is representative for the following reasons: Firstly, the applications could not just be tools or libraries, but had to be complete applications that demonstrate the benefits of RDF or Linked Data to the end-user of the application. Secondly, the applications submitted to the challenges already follow baseline criteria as required in [Klein & Visser, 2004], e.g. most of them use real-world data. Thirdly, the authors of each submitted application were also required to submit a scientific paper focusing on the implementation aspects of their application, which allows them to explain the goals of their application in their own terms. Finally, each challenge was held over multiple years, which makes it easier to compare the submissions from the different years in order to see trends, such as the uptake of the Linked Data principles.

The second threat to validity, is the number of authors who verified the data about their applications. Only 65% of authors replied to this request for validation. This may be due to the short-lived nature of academic email addresses. In several instances, we tried to find an alternative email address, if an address had expired. Every author that we successfully contacted validated the data about their application, which usually included only small corrections for one or two properties. We were unable to validate the data for authors that could not be reached. As the first challenge was already held in 2003, we do not believe that a much higher validation rate would have been possible at the time of writing.

The third threat to validity is that for each application only the associated academic paper and not the source code were analysed for the survey. There are several reasons for this. At the time of writing most demonstration prototypes, except those from the most recent years, were not deployed anymore. Publication of the source code of the applications was not a requirement for both challenges. In addition, it is very likely that most applications would not have been able to make their source code available due to IP or funding restrictions.

3.3. Empirically-grounded conceptual architecture

Based on our empirical analysis, we now present a conceptual architecture for Linked Data applications. As defined by [Soni et al., 1995], a *conceptual architecture* describes a system in terms of its major design elements and the relationships among them, using design elements and relationships specific to the application domain and use case. It enables the discussion of the common aspects of implementations from a particular application

domain, and can be used as a high-level architectural guideline when implementing a single application instance for that application domain.

Our conceptual architecture describes the high level components most often used among the surveyed applications to implement functionality that substantially differentiates applications using RDF or Linked Data from database-driven applications. For each component we describe the most common implementation strategies that are suited for specific application goals and deployment scenarios.

We first explain the choice of architectural style for our conceptual architecture, and we describe our criteria for decomposing the surveyed applications into components. Then we provide a detailed description of the components.

3.3.1. Architectural style of the conceptual architecture

[Fielding, 2000] defines an architectural style as a coordinated set of architectural constraints that restricts the roles and features of architectural elements and the allowed relationships among those elements within any architecture that conforms to that style.

Our empirical survey of RDF-based applications showed that there is a range of architectural styles among the surveyed applications, which is dependent on the constraints of the respective software architects. For example, as existing Web service infrastructure can be reused as part of an application, many applications chose a *service-oriented architecture*. Applications that are focused on reusing existing databases and middleware infrastructure preferred a *layered style*. The *client-server architecture* was used when decentralised deployment across organisations or centralised storage were important. Applications that prioritised robustness, decentralisation or independent evolution favoured a *peer-to-peer architecture*. Finally, the architecture of all other applications usually was defined by the reuse of existing tools and libraries as components in a *component-based architecture*.

As the component-based architectural style is the lowest common denominator of the surveyed applications, we chose the component-based architectural style for our conceptual architecture. It allows us to express the most common high-level functionality that is required to implement Semantic Web technologies as separate components.

3.3.2. Decomposing the surveyed applications into components

In order to arrive at our component-based conceptual architecture, we started with the most commonly found functionality amongst the surveyed applications: a component that handles RDF, an integration component and a user interface. With these components as a starting point, we examined whether the data from the architectural analysis suggested

Table 3.4.: Results of the architectural analysis by year and component

year	number of apps	graph access layer	RDF store	graph-based nav. interface	data homog. service	graph query language service	structured data auth. interface	data discovery service
2003	10	100%	80%	90%	90%	80%	20%	50%
2004	16	100%	94%	100%	50%	88%	38%	25%
2005	6	100%	100%	100%	83%	83%	33%	33%
2006	19	100%	95%	89%	63%	68%	37%	16%
2007	24	100%	92%	96%	88%	88%	33%	54%
2008	23	100%	87%	83%	70%	78%	26%	30%
2009	26	100%	77%	88%	80%	65%	19%	15%
total	124	100%	88%	91%	74%	77%	29%	30%

we split them into further components. Table 3.4 shows the results of this architectural analysis. The full data of the analysis is available online[4].

Every surveyed application (100%) makes use of RDF data. In addition, a large majority (88%), but not all surveyed applications implement persistent storage for the RDF data. In practice many triple stores and RDF libraries provide both functionality, but there are enough cases where this functionality is de-coupled, e.g. if the application has no local data storage, or only uses SPARQL to access remote data. This requires splitting of the RDF-handling component into two components. First, a *graph access layer*, which provides an abstraction for the implementation, number and distribution of data sources of an application. Second, an *RDF store* for persistent storage of graph-based RDF data.

A query service which implements SPARQL, or other graph-based query languages, in addition to searching on unstructured data, is implemented by 77% of surveyed applications. Such high coverage suggested the need for a *graph query language service*. It is separate from the graph access layer, which provides native API access to RDF graphs.

A component for integrating data is implemented by 74% of the surveyed applications. It provides a homogeneous perspective on the external data sources provided by the graph access layer. Usually external data first needs to be discovered and aggregated before it can be integrated - an integration service would offer this functionality. However only 30% of the surveyed applications required this functionality. For this reason, we split the integration functionality into a *data homogenisation service* and a *data discovery service*.

[4] http://preview.tinyurl.com/component-survey-results-csv

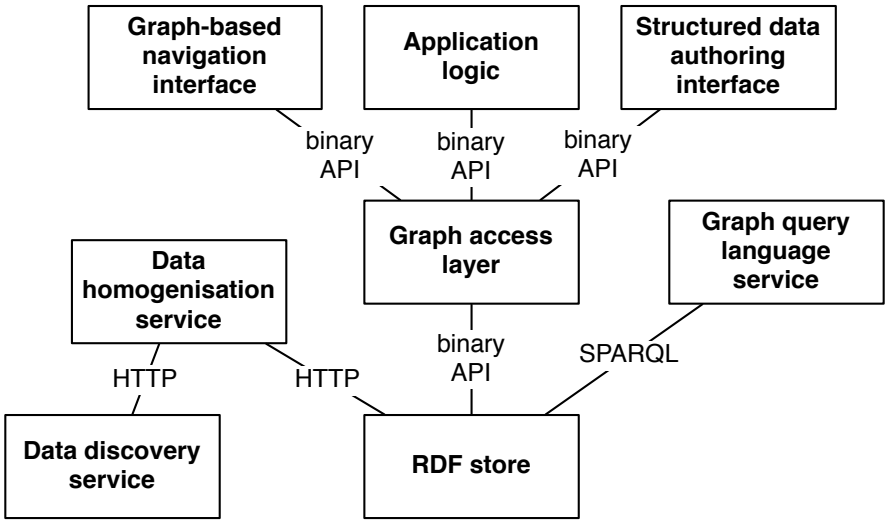

Figure 3.1.: Example component diagram showing a hypothetical application with all components.

Most (91%) of the surveyed applications have a user interface. All user interfaces from the surveyed applications allow the user to navigate the graph-based data provided by the application. Only 29% provide the means for authoring new data. Thus the user interface functions were split between two components: the *graph-based navigation interface* and the *structured data authoring interface*.

3.3.3. Description of components

The resulting seven components describe the largest common high-level functionality which is shared between the surveyed applications in order to implement the Linked Data principles and Semantic Web technologies. For each component we give a name, a description of the role of the component, and a list of the most common implementation strategies for each component.

Figure 3.1 shows the component diagram for the example application in section 3.3.4.

Graph access layer

Also known as data adapter or data access provider. This component provides the interface needed by the application logic to access local or remote data sources, with the distinction based on physical, administrative or organisational remoteness. In addition this component provides a translation or mapping from the native data model of the programming language to the graph-based data model of RDF. All (100%) of the surveyed applications have a graph access layer.

This component is separate and distinct from the RDF store and graph query language service components, as it provides an abstraction layer on top of other RDF stores, query interfaces and legacy document storage or relational databases.

Libraries for accessing RDF stores locally or remotely via SPARQL are available for all major programming and scripting languages. [Oren et al., 2008] describes the ActiveRDF approach for mapping between object oriented programming languages and the RDF data model. Another more recent approach to map between object oriented porgramming languages and the RDF data model is the Object triple mapping (OTM) described by [Quasthoff & Meinel, 2012].

Implementation strategies: Accessing local data is implemented via programmatic access through RDF libraries by at least 50% of the surveyed applications. 47% use a query language for accessing local or remote data sources. Most of these applications use the SPARQL standard. Decentralised sources are used by 85%; 91% use data from multiple owners; 73% can import external data and 69% can export their data or provide a SPARQL end-point to make their data reusable. 65% of the surveyed applications support data-updating during run-time.

RDF store

Also known as triple store, persistent storage or persistence layer. This component provides persistent storage for RDF and other graph based data. It is accessed via the graph access layer. 88% of surveyed applications have such functionality.

[Bizer & Schultz, 2009] provide an overview of the features and the performance of RDF stores as part of the Berlin SPARQL Benchmark results. Notable examples of RDF stores include the Sesame [Broekstra et al., 2002] and Jena [Wilkinson et al., 2003] stores.

Implementation strategies: 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 96% of applications; OWL is supported by 43%; RDF Schema is supported by 20%. Inferencing or reasoning on the stored data is explicitly mentioned by 58% of surveyed applications. Another implementation strategy is to use a relational database to store RDF data.

Data homogenisation service

Also known as integration service, aggregation service, mediation service or extraction layer. This component provides a means for addressing the structural, syntactic or semantic heterogeneity of data resulting from data access to multiple data sources with different formats, schemas or structure. The service goal is to produce a homogeneous

view on all data for the application. The data homogenisation service often needs to implement logic which is specific to a domain or an application. 74% of surveyed applications implement this component.

[Doan & Halevy, 2005] provides a general overview of semantic integration, and [cro, 2005] provides an overview of using Semantic Web technologies for integration.

Implementation strategies: Integration of heterogeneous data is supported by 87% of surveyed applications; 91% support data integration from sources with different ownership. Integration of data from distributed, decentralised data sources is supported by 85%. These three properties are orthogonal, as it would be possible, for example, to support just SIOC data, which is not heterogeneous, but which may be aggregated from personal websites, so that the data sources are distributed and under different ownership.

The four major styles of implementing the data homogenisation service are: Automatic integration (11%), which is performed using heuristics or other techniques to avoid human interaction. Semi-automatic integration (59%), which requires human inspection of the source data, after which rules, scripts and other means are manually created to integrate the data automatically in the future. Manual integration (10%) in which the data is completely edited by a human. Finally, 20% do not need any data integration because they operate on homogeneous sources.

Data discovery service

Also known as crawler, harvester, scutter or spider. This component implements automatic discovery and retrieval of external data. This is required where data should be found and accessed in a application domain specific way before it can be integrated. 30% of surveyed applications implement a data discovery service.

[Harth et al., 2010] introduce different research issues related to data crawling. [Käfer et al., 2013] provide an empirical survey of the change frequency of Linked Data sources.

Implementation strategies: The component should support 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 service can run once if data is assumed to be static or continuously if the application supports updates to its data (65%).

Graph query language service

Also known as query engine or query interface. This component provides the ability to perform graph-based queries on the data, in addition to search on unstructured

data. Interfaces for humans, machine agents or both can be provided. 77% of surveyed applications provide a graph query language service.

[Mangold, 2007] provides a survey and classification of general semantic search approaches. [Arenas & Pérez, 2011] provide an introduction to SPARQL and how it applies to Linked Data.

Implementation strategies: Besides search on features of the data structure or semantics, generic full text search can be provided. An interface for machine agents may be provided by a SPARQL, web service or REST endpoint. SPARQL is implemented by 41% of applications, while the preceding standards SeRQL and RDQL are explicitly mentioned by only 6%.

Graph-based navigation interface

Also known as portal interface or view. This component provides a human accessible interface for navigating the graph-based data of the application. It does not provide any capabilities for modifying or creating new data. 91% of surveyed applications have a graph-based navigation interface.

[Dadzie & Rowe, 2011] provide an overview of approaches used for visualising Linked Data. One of the most popular JavaScript libraries for displaying RDF and other structured data as faceted lists, maps and timelines is Exhibit [Huynh et al., 2007].

Implementation strategies: 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 an application domain specific visualisation, e.g. on a map. Most navigation interfaces allow the user to perform searches, [Hildebrand et al., 2007] provides an analysis of approaches for user interaction with semantic searching.

Structured data authoring interface

This component allows the user to enter new data, edit existing data, and import or export data. The structured data authoring component depends on the navigation interface component, and enhances it with capabilities for modifying and writing data. Separation between the navigation interface and the authoring interface reflects the low number of surveyed applications (29%) implementing write access to data.

The Semantic MediaWiki project [Krötzsch et al., 2006] is an example where a user interface is provided for authoring data. Another example is the OntoWiki [Heino et al., 2009] project.

Implementation strategies: The annotation task can be supported by a dynamic interface based on schema, content or structure of data. Direct editing of data using

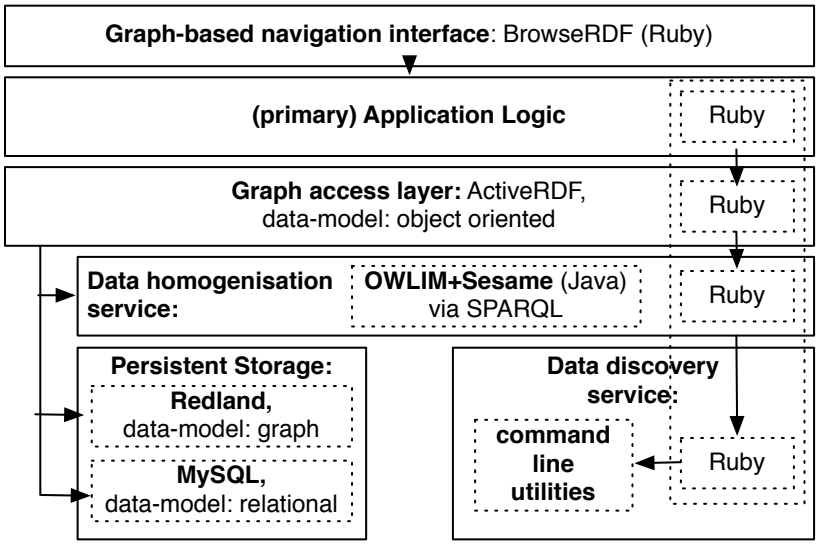

Figure 3.2.: Result of the architectural and functional analysis of an example application, the SIOC explorer

standards such as e.g. RDF, RDF Schema, OWL or XML can be supported. Input of weakly structured text using, for example, wiki formatting can be implemented. Suggestions for the user can be based on vocabulary or the structure of the data.

3.3.4. Analysis of an example application

In order to demonstrate how our conceptual architecture applies to an RDF-based application, we show the architecture of an application from the authors previous work, the SIOC explorer [Bōjars et al., 2007], in Figure 3.2.

The SIOC explorer, aggregates content from weblogs and forums exposing their posts and comments as RDF data using the SIOC vocabulary [Breslin et al., 2006]. This allows following multiple blogs or forums from a single application. The *application logic* and most parts of the application are implemented using the Ruby scripting language and the Ruby on Rails web application framework. The *graph-based navigation interface* allows faceted browsing of the SIOC data and is implemented through the BrowseRDF Ruby component [Oren et al., 2007]. The *graph access layer* is provided by ActiveRDF [Oren et al., 2008], which is an object-oriented Ruby API for accessing RDF data. The *data homogenisation service* is implemented in two steps: (1) generic object consolidation is provided by the OWLIM extension for Sesame[5], written in Java, and accessed via SPARQL, and (2) application domain specific integration of SIOC data is implemented as Ruby code. The Redland library[6] is used as an RDF store. Other application data (non-RDF data) is persistent to a MySQL relational database. A *data*

[5] http://ontotext.com/owlim/
[6] http://librdf.org/

discovery service is implemented through several Unix command line utilities which are controlled by Ruby. The SIOC explorer does not implement a graph query language search service or a structured data authoring interface.

3.4. Extending the proposed conceptual architecture for recommender systems

The proposed conceptual architecture has been derived from the surveyed 124 applications. The surveyed applications have been developed for many different application scenarios and areas. These areas include e.g. life sciences, geo-informatics, digital humanities, e-government, knowledge management, and social networking.

As such, the proposed conceptual architecture needs to be seen as a lowest common denominator, which only includes components required for generic Linked Open Data applications, which are not customised for any specific application area.

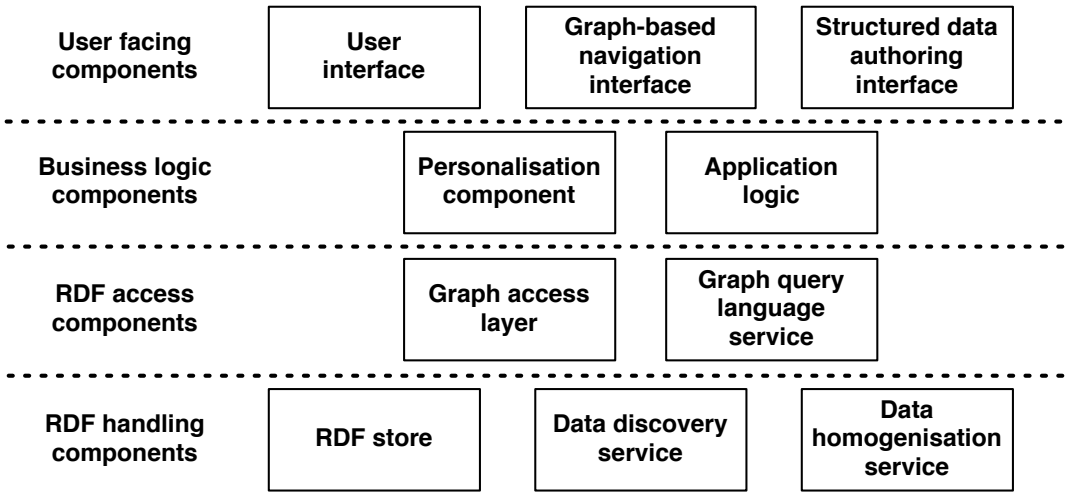

Figure 3.3.: Conceptual architecture for Linked Open Data recommender systems, as an extension of the proposed conceptual architecture for generic Linked Data applications.

In order to show how to use this generic conceptual architecture for recommender systems, we present an extend version of the conceptual architecture which is customised for recommender systems. We have extended the generic architecture with a *personalisation component* and a generic *user interface* component. Figure 3.3 shows all of the components of the extended conceptual architecture, grouped by component type.

User facing components: These are the components that implement user interfaces, using data provided by the lower components. We have added a generic *user interface*

component, as recommender systems do not necessarily need to provide any form of *graph based navigation interface* or *structured data authoring interface*.

Business logic components: These components implement the business rules that determine how data can be created, displayed, stored or changed. Here we distinguish between the *personalisation component*, which generates recommendations according to the recommendation algorithm and the business rules of the system, and the *application logic* component which implements un-personalised logic. Both the personalisation component and the application logic make use of the RDF access components in order to access their data.

RDF access components: These components enable access to RDF data in an abstract way. They provide an abstraction on top of the RDF handling components. The *graph access layer* provides a translation from the graph-based data model of RDF to the native model of the programming language. The *graph query language service* provides access via a graph query language such as SPARQL.

RDF handling components: These components directly store or process RDF data. The *RDF store* provides persistent storage of RDF data. The *data discovery service* handles discovery and retrieval of external data sources before integrating the data. The *data homogenisation service* integrates data from sources with different formats, schema or structure.

We instantiate this conceptual architecture for Linked Open Data recommender systems for our prototype in Chapter 6.

3.5. Related work

We now discuss related work in the areas of empirical surveys about the adoption of Semantic Web technologies and architectures for Semantic Web applications.

3.5.1. Empirical surveys

[Cardoso, 2007] presents the results of a survey of 627 Semantic Web researchers and practitioners carried out in January 2007. The questions from the survey covered the categories of demographics, tools, languages and ontologies. The goal of the survey was to characterise the uptake of Semantic Web technologies and the types of use-cases for which they were being deployed. Concrete applications were not taken into account. As the survey was carried out over a two months period of time, it does not allow conclusions about long term trends before or after the survey was taken.

Another similar survey of 257 participants (161 researchers and 96 industry-oriented participants) was published online[7] in 2009. The data for this survey was taken from a 3 month period. This survey had the goal of measuring the general awareness of Semantic Web technologies and social software in academia and enterprise. As such, the questions did not go into any implementation specific details.

[Cunha & de Lucena, 2006] performed a survey of 35 applications from the "Semantic Web challenges" in 2003, 2004 and 2005. The goal of the survey was to cluster the applications based on their functionality and their architecture. The result is a list of 25 application categories with the number of applications per category for each year. This survey covers only 3 years, and it is not concerned with the adoption of Semantic Web technologies and capabilities, whereas our empirical survey provides empirical data on the uptake of e.g. data integration or the Linked Data principles.

3.5.2. Architectures for Semantic Web applications

[Garcia-Castro et al., 2008] propose a component-based framework for developing Semantic Web applications. The framework consists of 32 components aligned on 7 dimensions, and is evaluated in 8 use-cases. While existing software tools and libraries that implement these components are identified, the identification of the components is not based on an empirical grounding.

[Tran et al., 2007] propose a layered architecture for ontology-based applications that is structured in a presentation layer, logic layer and data layer. It is based on best practices for service-oriented architecture and on the authors model of life-cycle management of ontologies, and evaluated in a use-case. However there is no empirical grounding for the architecture.

A similar approach is taken by [Mika & Akkermans, 2003] who propose a layered architecture for ontology-based applications, with layers for the application, the middleware and the ontology. The architecture is based on a requirements analysis of KM applications.

[Cunha & de Lucena, 2007] build on their earlier survey in [Cunha & de Lucena, 2006] and presents an UML architecture based on the 35 analysed applications. The goal of the architecture is to provide the foundation for a potential software framework, but no evaluation or implementation of the architecture is provided.

[7] http://preview.tinyurl.com/semweb-company-austria-survey

3.6. Summary

In this chapter, we have derived a conceptual architecture for Linked Open Data recommender systems. It is based on an empirical survey of RDF-based applications. The proposed conceptual architecture describes the components and their high-level functionality that a recommender system can implement in order to leverage Linked Open Data.

In order to provide a strong empirical grounding for the conceptual architecture, we first performed an empirical survey of 124 applications. The applications come from the years 2003 to 2009 of the "Semantic Web challenge" and the years 2006 to 2009 of the "Scripting for the Semantic Web challenge". The findings of our survey show that support for the graph-based data model of RDF is virtually at 100% amongst the surveyed applications, and most applications reuse formal semantics such as the FOAF, DublinCore and SIOC vocabularies. Support for the Linked Data principles has reached more then half of the contest entries in 2009. Support for decentralised sources, data from multiple owners, and sources with heterogeneous formats is supported each year by a large majority of the surveyed applications. In addition, the majority of applications leverage Semantic Web technologies to support data integration.

We then used the results of the empirical survey as the empirical foundation for a component-based, conceptual architecture for Linked Data applications. The proposed conceptual architecture describes the functionality which is required for a recommender system in order to leverage RDF, Linked Data and Semantic Web technologies. Table 3.5 shows a summary of the functionality of each component of the architecture.

The components of the proposed conceptual architecture allow implementing the functionality which is required in order to leverage Linked Open Data for personalisation in open ecosystems:

Multi-source user profiles are enabled by using the data discovery service component to aggregate distributed user profiles, and by using the data homogenisation service to integrate the different user profiles after they are aggregated. The resulting multi-source user profiles are then stored in the RDF store via the graph access layer or the graph query language service.

Domain-neutral user profiles are enabled by using the graph-based data model of RDF for representing the user profile, and by using concept identifiers from Linked Open Data to represent the user preferences from the different domains. The resulting domain-neutral user profile can then be stored in the RDF store via the graph access layer or the graph query language service.

Cross-domain personalisation is enabled by using an approach which can leverage the graph-based data model of RDF. The personalisation component then executes the

Table 3.5.: Summary of the proposed conceptual architecture for Linked Open Data recommender systems

Component	Summary of functionality
User interface	Generic user interface without graph-based features or authoring capabilities.
Graph-based navigation interface	Human accessible interface for navigating and viewing graph-based data.
Structured data authoring interface	Allows the user to enter new data and/or edit existing data.
Personalisation component	Provides environment for running the recommendation algorithm. Provides user profiles and background data, and delivers results to user facing components.
Application logic	Implements the application logic for creating, displaying, storing or changing data.
Graph access layer	Abstraction layer, provides translation from graph-based data model of RDF to the native model of the programming language.
Graph query language service	Allows performing graph-based queries and/or general searches.
RDF store	Persistent storage for graph-based RDF data.
Data homogenisation service	Provides homogeneous view on data from multiple sources. Integrates heterogeneous data from sources with different formats, schema or structure.
Data discovery service	Discovery and retrieval of external data sources before integrating the data.

recommendation algorithm, and uses the graph access layer to access background knowledge and the domain-neutral user profiles from the RDF store. The resulting recommendations are then passed on to the user interface or the graph-based navigation interface.

3.6.1. Contributions of this chapter

In this chapter, we have derived a conceptual architecture for Linked Open Data recommender systems. It is based on an empirical survey of RDF-based applications. The

proposed conceptual architecture describes the components and their high-level functionality that a recommender system can implement in order to leverage Linked Open Data.

In this chapter we have addressed research question (Q5):

Research question (Q5): Which components and best practices need to be implemented by a recommender system in order to leverage Linked Data for recommendations?

The proposed conceptual architecture for Linked Open Data recommender systems provides a template and best practices for the typical high level components required for providing personalisation in open ecosystems. Prospective developers can use it as a guideline when designing and implementing a recommender system that uses Linked Open Data to enable cross-domain recommendations using multi-source and domain-neutral user profiles.

In Chapter 6, we will describe the implementation of our cross-domain personalisation prototype. The architecture of the prototype is based on the presented conceptual architecture, and it implements most of the components of the architecture in order to leverage Linked Data for the cross-domain recommendation task.

Chapter 4.

Cross-Domain Personalisation Algorithm: SemStim

In this chapter, we introduce SemStim, our algorithm for cross-domain personalisation. SemStim is an unsupervised, graph-based recommendation algorithm that uses Linked Data from DBpedia in order to provide recommendations. It is an enhanced version of spreading activation (SA) as described by Crestani in [Crestani, 1997]. Spreading activation enables finding related entities in a semantic network, while still being deterministic and taking the semantics of the network into account. We extend spreading activation for the purpose of cross-domain recommendation, by allowing the definition of sets of activation targets and by introducing a way to control the duration of the algorithm.

Our algorithm is innovative due to the main design choices of our approach:

1. Our algorithm has been designed to generate recommendations in a target domain based on user interests in a different source domain, by using indirect connections from DBpedia between items of the two domains. This enables us to provide *cross-domain recommendations*, even in the worst-case scenario, when there is no overlap between users and items of the two domains.

2. It is not affected by the *cold-start problem*, as it only requires user preferences of the user for whom recommendations are being generated. In other words, it will even work when there is only a single user in the system.

Contributions of this chapter: We address research question (Q4) in this chapter. (Q4) asks for a recommendation algorithm that can generate cross-domain recommendations for two different domains without overlap. As we will show, our algorithm provides this capability, as it enables personalisation using multi-source and domain-neutral user profiles. Such user profiles are provided by open ecosystems in which user preferences can be shared between recommender systems, as introduced in the background chapter (Chapter 2).

Chapter outline: In Section 4.1, we describe the details of our SemStim algorithm. We first describe the intuition behind the algorithm, after which we provide a formal description of the algorithm.

This is followed by a discussion of other algorithms which are related to SemStim in Section 4.2. First we discuss both the common aspects and the differences between SemStim and SA as described by [Crestani, 1997]. Then we describe the differences between the non-linear activation model of SemStim and the linear activation model which is frequently used by other formalisations of SA. Then we discuss the relation between PageRank and SA, and between artificial neural networks and SA.

4.1. Algorithm Description

SemStim extends the spreading activation (SA) algorithm as described by [Crestani, 1997] for the purpose of cross-domain recommendation. We add (1) *targeted activation*, which allows us to describe the target domain of the personalisation, and (2) we add a set of *constraints* to control the algorithm *duration*.

We first provide an outline of the algorithm to explain the intuition behind the algorithm and its formalisation. Then we provide a formal description of the SemStim algorithm. This followed by a comparison and discussion of related algorithms.

4.1.1. Intuition behind the SemStim algorithm

We first describe the intuition behind the algorithm, without going into any formal details, such as the different variables involved in the algorithm.

SemStim is an iterative algorithm. The input of the algorithm are the nodes in the user profile from the source domain. Each of these nodes will spread an amount of activation to his direct neighbors in the very first iteration. Whenever an unactivated node has accumulated more activation then its threshold, it becomes activated. In each subsequent iteration, all nodes that became activated in the current iteration spread an amount of activation to their neighbours. This in turn might lead to other nodes becoming newly activated, which will spread activation in the following iteration.

When enough nodes from the target domain have become activated, the algorithm terminates. The activated nodes from the target domain are returned as recommendations, ranked by their activation level. Nodes which are not from the target or source domain, can become activated as well. These nodes allow traversing the parts of the graph which indirectly connect the source and target domain.

Note that SemStim introduces differences to basic spreading activation, as will become apparent in the formal description of the algorithm, and our discussion of the differences between SemStim and spreading activation. Most importantly, SemStim introduces constraints to control the algorithm duration. Each iteration of the algorithm is called a *wave*. The algorithm can be restarted if it terminates before reaching the required number of activated nodes from the target domain. Each restart is called a *phase*. By specifying the maximum number of waves and phases, the algorithm duration can be controlled.

In addition, it is important to note that for the SemStim algorithm each node can only fire under two conditions: (1) when it becomes activated the very first time, and (2) when the algorithm restarts and the node was already activated in the previous phase. As we will discuss in Section 4.2.2, there are other formalisations of SA in which each activated node fires in each iteration/wave.

4.1.2. Formal description of the SemStim algorithm

Activation is spread on a graph that uses the RDF data model, so we define the graph $G = (V, E, T)$, with V as the set of nodes $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$ as the set of predicates used as edges in the graph. Then T is the set of RDF triples in the graph, with $T \subseteq V \times E \times V$.

We interpret the RDF data model[1] as follows:

RDF URIs: The set of URIs in the graph is $U = V \cup E$, as URIs can be used as subject, predicate and object in RDF triples. Note that for the purpose of formalising the SemStim algorithm, we still distinguish between V as the set of nodes (or vertices), and E as the set of edges, although both sets only contain RDF URIs.

RDF literals: We exclude RDF literals as they can only be used as objects in RDF triples. Therefore RDF literals are sinks for spreading activation throughout the graph, as no activation can spread from them.

RDF blank nodes: We also exclude RDF blank nodes, as the recommendations for publishing Linked Data [Bizer et al., 2007] specify to not use them in Linked Data.

Direction of links in the RDF graph: We view the RDF graph as undirected. While the RDF data model specifies that all edges in an RDF graph are directed, we have found that following edges only in one direction can exclude parts of the DBpedia graph from the spreading of activation, as e.g. all DBpedia categories are always connected to their entities in the same direction.

[1] http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

Table 4.1.: Summary of functions which are part of the SemStim algorithm

Function	Description
Input function $I(v, w)$	Aggregates the weighted output of the direct neighbors of node.
Activation function $A(v, w)$	Determines the amount of activation which a node can spread to its neighbors when it reaches its activation threshold.
Output function $O(v, w)$	Determines the initial output of a node before applying modifiers like the degree penalty.
$Fire(v, w)$ function	Determines if a node can fire in wave w.
$Restart(\mathbf{a}^{(w)})$ function	Determines if the algorithm can be restarted, if it terminated without activating the required number and type of nodes.
$Terminate(\mathbf{a}^{(w)})$ function	Determines if the algorithm as a whole can terminate, because the required number and type of nodes are activated.

`rdf:type` links: In addition, we exclude all `rdf:type` links. These links connect URIs to their RDF classes. RDF classes have a very high centrality in the graph, because all the instances of an RDF class need to be connected to the corresponding class. This leads to the RDF class nodes skewing the results of the algorithm, by appearing on all results irrespective of the starting nodes.

We denote each iteration of the algorithm as *wave* $w \in \mathbb{N}_0$, and each restart of the algorithm as a *phase* $p \in \mathbb{N}_0$. The maximum number of phases is given by ρ_{max} .

The *activation state* of the graph G in wave w is denoted by $\mathbf{a}^{(w)} \in \mathbb{R}^n$, with $\mathbf{a}_v^{(w)}$ as the activation of node $v \in V$.

For the purpose of the formal description of SemStim, we use the following definitions of source and target domain: The *source domain* $S = \{s_1, \ldots, s_f | s_i \in V\}$ is the subset of V from which items in the user profiles are taken, e.g. all MovieLens movies on DBpedia. The *target domain* $D = \{d_1, \ldots, d_e | d_i \in V\}$ is the set of nodes which represent the recommendable items among all of the nodes in V, e.g. all books on DBpedia. Each *user profile* P_u is a set of nodes $P_u = \{p_1, \ldots, p_k | p_i \in S\}$ from the source domain S. Then P_u denotes the user preferences of user u.

Table 4.2.: List of variables used by the SemStim algorithm

Variable	Description
τ	The activation threshold of all nodes.
β	The default weight of all predicates. Alternatively, a different weight $\beta_i := \beta(p_i)$ can be assigned per predicate p_i.
α	The initial output of a node before applying modifiers like the degree penalty.
w_{max}	The maximum number of waves. Each wave is an iteration of the algorithm.
ρ_{max}	The maximum number of phases. Each restart of the algorithm marks a new phase.
θ	The required number of activated nodes from the target domain D. After activating this number of nodes the algorithm will terminate.

As SemStim does not make use of user preferences in the target domain, we will not include them in the formalisation of SemStim. Further, without loss of generality, we will assume that SemStim is used for one source domain and one target domain. If more then one source domain is used, then the items in all source domains can be merged into one source domain. The same applies to the target domain.

We have summarised the list of variables used by the SemStim algorithm in table 4.2. In addition, table 4.1 shows a summary of the functions used by the algorithm.

The *initial activation state* $\mathbf{a}^{(0)}$ is defined by setting the initial activation of all nodes in the user profile P_u to the same value. τ is the activation threshold of a node, which determines how much activation a node needs to accumulate before it can fire. Setting the value of τ influences the accuracy and the diversity of the recommendations generated by SemStim. We describe experiments, which compare the recommendations of SemStim for different values of τ as part of our evaluation. Section 5.5 describes an experiment in which the value of τ influences the diversity of the recommendations. Section 5.6 describes the results of an experiment in which the diversity of the results of SemStim is inversely correlated to the accuracy of the results.

We set the initial activation of all nodes in P_u to τ, so that these nodes can fire and contribute to the activation state $\mathbf{a}^{(1)}$ in wave 1.

$$\mathbf{a}_v^{(0)} = \begin{cases} \tau & ,\forall v \in P_u \\ 0 & ,\forall v \in V \setminus P_u \end{cases} \tag{4.1}$$

For each iteration/wave w of the algorithm, the activation state $\mathbf{a}_v^{(w)}$ for a node v with $w > 0$ is obtained from the previous state $\mathbf{a}_v^{(w-1)}$ by first applying the input function $I(v, w)$ to v. Then the activation function $A(v, w)$ is applied to the result. As the last step, the output function $O(v, w)$ is applied to the result of $A(v, w)$, as specified below.

The *input function* $I(v, w)$ aggregates the weighted output of the direct neighbors of node v in wave w.

$$I(v, w) = \mathbf{a}_v^{(w)} = \mathbf{a}_v^{(w-1)} + \sum_{\substack{s \in V,\ p \in E \\ s \neq v}} \sum W(s, p, v)\, O(s, w-1) \tag{4.2}$$

$$W(s, p, o) = \begin{cases} \beta & ,\text{if } (s, p, o) \in T \vee (o, p, s) \in T \\ 0 & ,\text{otherwise} \end{cases} \tag{4.3}$$

The weight of each triple (s, p, o) is defined by $W(s, p, o)$. The direction of an edge in the graph is not taken into account, resulting in an undirected view on the graph. β is the standard weight of all edges in the graph. In our experiments, we set $\beta = 1.0$. The main benefit of using a uniform weight, is that it makes SemStim an unsupervised approach, which can be used for any combination of domains, without first requiring to learn weights depending on the choice of target and source domain.

However it is possible to make the value of β dependent on the predicate, by substituting $\beta_i := \beta(p_i)$ in order to give different weights to different predicates $p_i \in E$. This allows SemStim to take the semantics of different edge types into account. However, this would require a machine learning approach to learn the weight of different edge types, which goes beyond the scope of this thesis.

The *activation function* $A(v, w)$ determines the amount of activation which a node can spread to its neighbors when it reaches its activation threshold.

$$A(v, w) = \begin{cases} \alpha \frac{1}{Degree(v)} & ,\text{if } \mathbf{a}_v^{(w)} \geq \tau \\ 0 & ,\text{otherwise} \end{cases} \tag{4.4}$$

$Degree(v)$ is degree of node v, which is the combined number of in- and out-links of v, as we view the graph as undirected. α is the initial output when a node fires before taking other modifications like the degree penalty into account. Note that it is possible to use

non-linear functions to penalise the degree, and that additional penalties can be added, such as a penalty which increases with the distance of a node to any of the start nodes.

The *output function* $O(v, w)$ determines if a node can fire in wave w. This is dependent not just on the activation function $A(v, w)$, but also on binary functions which determine if the node can fire, or if the algorithm has reached the termination condition.

$$O(v, w) = A(v, w)\, Fire(v, w)\, Terminate(\mathbf{a}^{(w)}) \tag{4.5}$$

For the $Restart(\mathbf{a}^{(w)})$ and $Terminate(\mathbf{a}^{(w)})$ functions, we define $\tilde{a}^{(w)}(S)$ as the number of nodes in a given set S which have reached the activation threshold.

$$\tilde{a}^{(w)}(S) := \left|\{\mathbf{a}_v^{(w)} > \tau \mid v \in S\}\right| \tag{4.6}$$

The $Restart(\mathbf{a}^{(w)})$ function determines if a restart is necessary. Whenever the number of activated nodes stays the same between wave $w-1$ and wave w, then no further activation is possible without firing all activated nodes again. The $Restart(\mathbf{a}^{(w)})$ function checks this condition, and returns 1 if a restart is necessary. Each restart marks the start of a new *phase* of the algorithm. Note that $Restart(\mathbf{a}^{(w)})$ takes the state $\mathbf{a}^{(w)}$ of all nodes into account.

$$Restart(\mathbf{a}^{(w)}) = \begin{cases} 1 & \text{, if } \tilde{a}^{(w)}(V) = \tilde{a}^{(w-1)}(V) \\ 0 & \text{, otherwise} \end{cases} \tag{4.7}$$

The $Fire(v, w)$ function determines if a specific node v in wave w can spread the result of its activation function to its neighbours. There are 2 conditions for firing: (1) The accumulated activation of the node was below the activation threshold τ in the previous wave $w-1$, and has reached τ in the current wave w. (2) If the $Restart(\mathbf{a}^{(w)})$ function signals a restart, and if the total number of restarts is below or equals the maximum number of phases ρ_{max}. Note that each node can only cross the activation threshold once, while it can be restarted up to ρ_{max} times.

$$Fire(v, w) = \begin{cases} 1 & \text{, if } \mathbf{a}_v^{(w-1)} < \tau \wedge \mathbf{a}_v^{(w)} \geq \tau \\ 1 & \text{, if } Restart(\mathbf{a}^{(w)}) = 1 \\ & \quad \text{and } (\sum_{k=0}^{w} Restart(\mathbf{a}^{(k)})) \leq \rho_{max} \\ 0 & \text{, otherwise} \end{cases} \tag{4.8}$$

The $Terminate(\mathbf{a}^{(w)})$ function determines if the algorithm as a whole can terminate. There are two termination conditions: (1) the number of activated nodes from the target domain D is higher then required number of targets, which is given as θ. Or (2) if the algorithm has reached the maximum number of waves, which is specified by w_{max}. If

neither of the two termination conditions is fulfilled, then the function returns 1, which allows nodes to spread activation, should they fire. However, if one termination condition is fulfilled, then the function returns 0, which suppresses all nodes from further spreading any activation, thus terminating the algorithm. Once this happens, the algorithm has terminated, and the activation state $\mathbf{a}^{(w)}$ will not change.

$$Terminate(\mathbf{a}^{(w)}) = \begin{cases} 0 & \text{, if } \tilde{a}^{(w)}(D) > \theta \\ 0 & \text{, if } w > w_{max} \\ 1 & \text{, otherwise} \end{cases} \tag{4.9}$$

We can summarise transition $\mathbf{a}^{(w)} \rightarrow \mathbf{a}^{(w+1)}$ for a node $v \in V$ through the sequential application of the input function $I(v, w)$, activation function $A(v, w)$ and output function $O(v, w)$ as follows:

$$\forall v \in V : \mathbf{a}_v^{(w+1)} = \mathbf{a}_v^{(w)} + Terminate(\mathbf{a}^{(w)}) \sum_{\substack{s \in V, \\ s \neq v}} \sum_{p \in E} W(s, p, v) A(s, w) Fire(s, w) \tag{4.10}$$

The *top-k recommendations* for $k \leq \theta$, user profile P and target domain D are determined after the termination by sorting the set $\{\mathbf{a}_v^{(w)} > \tau \mid v \in D\}$ of all activated nodes from the target domain by their activation values, and returning the first k items. Then the first k items are the nodes in the target domain that are most similar to the nodes in the user profile. We go into more detail related to the top-k recommendation task in Section 2.1.2.

Complexity of the algorithm

The computational complexity of the SemStim algorithm is $O((|E| \log(|V|))^{\rho_{max}})$, where $|E|$ is the number of edges in the graph, $|V|$ is the number of vertices and ρ_{max} is the maximum number of phases.

As described by [Rocha et al., 2004], basic spreading activation has a complexity of $O((|E| \log(|V|)))$. However, SemStim allows restarting the algorithm up to ρ_{max} times, which is the maximum number of phases. Therefore, for each phase has the complexity of a single basic spreading activation run, this resulting on an overall computational complexity of $O((|E| \log(|V|))^{\rho_{max}})$.

4.2. Related algorithms

Our proposed SemStim algorithm is superficially similar to some other graph-based algorithms. In this section, we discuss these related algorithms and describe the differences that make these related algorithms unsuitable for the task of cross-domain recommendations using Linked Data from DBpedia.

First we will discuss both the common aspects and the differences between SemStim and Spreading Activation (SA) as described by [Crestani, 1997], which is the most cited description of SA in the field of computer science. However, Crestani does not provide a full formal description of the SA algorithm. This leaves the formal details open for interpretation.

One of the most used formalisations of SA describes the algorithm using a linear activation model. This formalisation is used e.g. by [Berthold et al., 2009]. However, this introduces the problem of query independent termination states for the algorithm, which can lead to all users having the same recommendations. We show how SemStim avoids query independence through its non-linear activation model.

We conclude the discussion of related algorithms by discussing the relation between PageRank and SA, and by comparing artificial neural networks to SA.

4.2.1. Comparison between SemStim and Spreading Activation

SemStim is derived from the Spreading Activation (SA) algorithm. Spreading Activation is based on previous work in the 80's on semantic memory and case semantics [Cohen & Kjeldsen, 1987]. It is inspired by the fact that human memory retrieves memories by association. By recalling one situation, other memories which are strongly associated or connected are also remembered.

At the time of writing, the most cited description of SA in the field of computer science is published by [Crestani, 1997]. However, research about SA has also been published in other fields, such as psychology and linguistics. Crestani distinguishes between "pure spreading activation" and "constrained spreading activation".

Pure spreading activation

"Pure" Spreading Activation, is the most basic version of Spreading Activation which does not incorporate any kind of constraints for spreading across the semantic network. The algorithm consists of multiple iterations which are executed on the given semantic network, starting with a set of start nodes. The set of nodes which are activated after reaching the termination condition represent the most relevant nodes.

Each pulse of the spreading phase can activate new nodes. The spreading of the activation is determined by three functions of a node:

Input function: The total input of a node is determined by the input function: $I_j = \sum_i O_k w_{ij}$, where I_j is the total input of node j; O_i is the output of node i connected to node j; and w_{ij} is the weight associated to the link which connects node i and node j.

Activation function: The activation function determines if a node becomes activated, based on the amount of activation it has received. It is most often expressed as a threshold that is usually the same for each node. However it can also be set per node type or for individual nodes. If the input of the node is higher than the threshold, then the node is active in the next pulse of a spreading phase.

Output function: The output function determines how much output the node passes to all directly connected nodes in the next pulse. The output function can either produce a real value, which usually is the level of activation. Alternatively, the output function emits a binary value depending on its activation.

In addition, a global *termination condition* decides if the algorithm terminates or continues with its next iteration.

Each iteration consists of two steps:

1. one or more pulses, which are propagated to all nodes which are directly connected to an active node
2. a check of the termination condition, usually an upper limit to the number of activated nodes

Each pulse in turn is made up of three phases:

1. pre-adjustment
2. spreading
3. post-adjustment

Crestani does not provide any further formal details about the iterations, pulses and phases of the algorithm. Neither are any details provided about the pre- and post-adjustment phases.

Constrainted Spreading Activation

In addition to the "pure" spreading activation algorithm, Crestani suggests four constraints when spreading the activation through the network. These constraints allow limiting the spreading activation to a subset of the graph, based on the distance to the start nodes,

the degree of a node, the type of the path connecting the node to any of the start nodes and the number or type of activated nodes.

distance constraint: the shortest distance between a node and any start nodes, can be used as a constraint in order to limit the activation from spreading to far away from the user profile.

fan-out constraint: nodes with a high number of out-links (a high fan-out) have a very broad semantic meaning. In most use cases the spreading activation should not follow such nodes, as this will lead to the activation of nodes with a weak connection to the initially activated nodes.

path constraint: certain link types can be preferred for the activation while other link types can be ignored for the activation.

activation constraint: different nodes can be more important then other nodes, e.g. to represent their importance to the application or the specific user. This can be taken into account by terminating the algorithm when the required type and number of nodes has been activated.

While these constraints appear reasonable and intuitive, no formal definition or implementation detail is provided by Crestani.

Differences between SemStim and standard Spreading Activation

While SemStim has been derived from Spreading Activation as presented by Crestani, it contains modifications and additions in order to make it suitable to the task of cross-domain recommendation on Linked Data from DBpedia.

SemStim is a constrained version of Spreading Activation, as we formally specify how to implement all four activation constraints which are described by Crestani. However, we do not make use of the path and distance constraints for reasons described below.

- The fan-out constraint is implemented as part of the activation function $A(v, w)$, by using the degree of a node as a penalty when spreading activation.
- The distance constraint can also be implemented as part of the activation function $A(v, w)$, by adding a penalty which is based on the distance of a node from any of the start nodes. However, we do not use the distance constraint for SemStim, because it can make it difficult to find the required number of nodes in the target domain.
- The activation constraint is implemented as part of the $Terminate(\mathbf{a}^{(w)})$ function, by checking if the number of activated nodes from the target domain has reached the target number θ.

- The path constraint can be implemented as part of the input function $I(v, w)$, by making the weight $\beta_i := \beta(p_i)$ of each triple (s, p_i, o) dependent on the predicate p_i. However, we did not implement the path constraint for SemStim, as we discovered that properties are not assigned to entities on DBpedia in a consistent way, so that e.g. the number of properties which are shared between all movie instances on DBpedia is very small. On the other hand, there is a large number of properties that are only used by three movies or less. Even intrinsically similar entities, like some of the Star Wars movies, do not share any properties.

In addition, our formalisation of SemStim has the following new features:

- We specify how to formally interpret the RDF data model for Spreading Activation. Spreading Activation requires a form of Semantic Network, however no formal details of such a semantic network are introduced by Crestani.
- We describe how to use the activation constraint in order to implement targeted activation by specifying the target domain D and the target number θ of activated nodes from D. We check θ in the $Terminate(\mathbf{a}^{(w)})$ function.
- We introduce constraints on the duration of the algorithm, by distinguishing between iterations in terms of phases and waves. Crestani only has the concept of a "pulse", which for him is a synonym for a single iteration. We use waves to refer to "naturally occurring" iterations of the SemStim algorithm, which occur when one or more nodes become activated after reaching their activation threshold. These nodes can then spread their activation in the next iteration. As soon as no nodes become newly activated, no further spreading of activation is possible without restarting the algorithm. We refer to a restart as a phase. At the start of a phase (beyond the initial phase), all activated nodes will spread their activation again. The maximum number of waves w_{max} is checked in the $Terminate(\mathbf{a}^{(w)})$ function. The maximum number of phases ρ_{max} is used in the $Fire(v, w)$ and $Restart(\mathbf{a}^{(w)})$ functions.

4.2.2. Comparison between the SemStim activation model and the linear activation model

As we discussed in Section 4.2.1, Crestani does not provide a full formal description of Spreading Activation. As a result, the choice of the input, activation and output function is left open. In this section, we compare the SemStim activation model to a popular alternative activation model, which is called the "Linear Activation Model". As we will explain, the Linear Activation Model leads to recommendations which are independent of the actual user profiles. For this reason, SemStim uses a non-linear activation model.

According to [Berthold et al., 2009], the most common way to instantiate the input, activation and output functions of Spreading Activation, is by choosing a linear input

function and by choosing the identity function for the activation function and the output function.

The identity function (see Equation 4.11) always returns the same value that was used as its argument.

$$id(x) = x \tag{4.11}$$

[Berthold et al., 2009] call such an activation model, in which a linear function is used as the input function, and in which the identity function is used for the activation and output functions, the "Linear Standard Model".

When using this choice of functions, the spreading activation process can be described using matrix notation. Given an activation vector $\mathbf{a}^{(k-1)}$, then the next iteration can be expressed as $\mathbf{a}^{(k)} = W\mathbf{a}^{(k-1)}$, where W is the weighted adjacency matrix, with $w(u, v) = 0$, if $(u, v) \notin E$.

We introduced the activation vector $\mathbf{a}^n$ in our formal description of SemStim in Section 4.1.2.

If the linear activation model is used, then this can be simplified as

$$\mathbf{a}^{(k)} = W^k\mathbf{a}^{(0)} \tag{4.12}$$

Equation 4.12 shows that Spreading Activation using linear activation without any other constraints corresponds to power iteration with matrix W. According to [Berthold et al., 2009], such a power iteration converges to the principal eigenvector of W, if the graph is connected and not bipartite. So after a sufficient number of iterations, the activation vector will approach eigenvector centrality, independent of the query. In terms of spreading activation, this means that the linear activation model will converge to eigenvector centrality independent of the starting nodes in the profile of the user.

In order to avoid *query independence* for SemStim, we do not use the linear activation model. Instead, SemStim has two features which are used to make the recommendations specific to each user profile:

1. We implement activation constraints as described in section 4.2.1.
2. We use our input, activation and output functions to implement a non-linear activation model.

While the SemStim input function $I(v, w)$ aggregates the incoming activation in a linear way, both the activation and output function are neither linear nor based on the identity function. The activation function $A(v, w)$ does not use the activation value of a node, instead the variable α is used as the initial output for the activation function.

Further, the degree of a node and optionally the distance of the node to one of the start nodes can be used as penalties in the activation function. Finally, our output function $O(v, w)$ is based on the binary functions $Fire(v, w)$ and $Terminate(\mathbf{a}^{(w)})$. As a result, instead of spreading their activation in each iteration of SemStim, nodes only can fire under 2 conditions: if they reach their activation threshold or if the algorithm gets restarted.

Taken together, the implementation of the activation constraints and the non-linear nature of the activation and output functions of SemStim avoid the query independence of the Linear Activation Model, and allow SemStim to provide personalised recommendations for each individual user.

4.2.3. Comparison between PageRank and SemStim

PageRank [Page et al., 1999] is a graph-algorithm, which is superficially similar to SemStim, as both algorithms are iterative algorithms in which nodes transfer a value to their neighbours. PageRank is an iterative algorithm in which nodes transfer a part of their own PageRank value to all the targets of their outbound links. However, as we will discuss, the results of the PageRank algorithm are query independent, which makes PageRank unsuitable for personalisation.

For the most basic form of the PageRank algorithm, the PageRank value of any node n, can be expressed as follows, where $Incoming(n)$ is the set of all nodes which have a link going into node n, and $OutDegree(m)$ is the number of links going out of node m.

$$PageRank(n) = \sum_{m \in Incoming(n)} \frac{PageRank(m)}{OutDegree(m)} \tag{4.13}$$

This calculation is performed for each node in the network in each iteration of the algorithm.

PageRank has additional features such as the damping factor and teleportation. PageRank is motivated by the idea that an imaginary surfer of the World Wide Web is randomly clicking links, to get from one web site to another. The dampening factor refers to the probability that the imaginary surfer will continue at any iteration of the algorithm. Teleportation refers to a transition between nodes which are not connected in the graph. If a node has no outgoing links, then it normally would terminate the algorithm. To avoid this, when the imaginary surfer arrives at such a sink page, he will continue randomly at any other node, with the probability being equally distributed between all nodes in the network.

The PageRank algorithm will converge on the principal eigenvector of the weighted adjacency matrix of a network, independent of additional features such as the dampening factor and teleportation.

As described in the previous Section 4.2.2, this is the same behavior as spreading activation without constraints when using the linear activation model. Therefore PageRank is query independent, which makes it unsuitable for personalisation.

4.2.4. Comparison between artificial neural networks and SemStim

The last related approach which we discuss, are artificial neural networks (ANN) [Yegnanarayana, 2009].

An artificial neural network is a network of nodes and connections between the nodes. The nodes are usually arranged in multiple layers, e.g. one layer of input nodes, one layer of hidden nodes, and a layer of output nodes. In order to form a network, each node of one layer is then connected to all the nodes of another layer. By using this kind of network topology, the ANN is trying to simulate the topology of neurons in a mammal brain. Each node in turn is simulating a neuron.

Nodes in an ANN are defined through three functions: an input function, an activation function and an output function. In regards to this, ANNs are similar to Spreading Activation and SemStim

However, the main difference between ANNs and Spreading Activation is in the network topology. The network topology of an ANN is completely artificial, as each node of one layer has outgoing connections to all nodes from another layer. The ANN then gets trained for a specific task by manipulating the weights between the nodes. In contrast, Spreading Activation and SemStim operates on a semantic network in which the network topology is induced by the semantic relationships between the concepts which are represented by the nodes in the network.

Due to this difference, the semantic network on which SemStim is operating gives a “meaning” to the concepts from different domains and their indirect connections. This in term allows SemStim to be used for cross-domain recommendations, whereas ANNs neither allow representing the concepts from different domains nor their indirect connections.

4.3. Summary

In this chapter, we have formally introduced SemStim, a spreading activation algorithm tailored to the RDF data model that can be used for cross-domain recommendation. It

leverages multi-source, domain-neutral user profiles and the semantic network of DBpedia in order to generate recommendations for different source and target domains. The algorithm requires that items are linked to a semantic network using the RDF graph data model. Beyond that it is not affected by the cold-start problem, or by the availability or quality of meta-data about recommendable items.

We first introduced the intuition behind the algorithm, then we described the formal details of the algorithm. We have summarised the list of variables used by the SemStim algorithm in table 4.2. In addition, table 4.1 shows a summary of the functions used by the algorithm.

In formal terms, we can summarise the transition of the activation state $\mathbf{a}^{(w)} \rightarrow \mathbf{a}^{(w+1)}$ for any node $v \in V$, with V as the set of all vertices of the graph, from wave w to wave $w+1$, through the sequential application of the input function $I(v,w)$, the activation function $A(v,w)$ and the output function $O(v,w)$, as follows:

$$\forall v \in V : \mathbf{a}_v^{(w+1)} = \mathbf{a}_v^{(w)} + Terminate(\mathbf{a}^{(w)}) \sum_{\substack{s \in V,\, p \in E \\ s \neq v}} \sum W(s,p,v) A(s,w) Fire(s,w)$$

The computational complexity of the SemStim algorithm is $O((|E| \log(|V|))^{\rho_{max}})$, where $|E|$ is the number of edges in the graph, $|V|$ is the number of vertices and ρ_{max} is the maximum number of phases. Each phase is a restart of the algorithm. The high complexity of the algorithm can be explained by the ability to restart the algorithm and by the non-linear activation model. Restarting the algorithm if the required number of target activations has not been reached, allows exploring a larger subset of the graph, which however also increases the complexity. The non-linear activation model is induced by our choice of input, activation and output function, and allows SemStim to provide personalised results.

The main difference between SemStim and basic SA as described by [Crestani, 1997], is that we extend SA for the purpose of cross-domain recommendation, by allowing the definition of sets of activation targets and by introducing waves and phases to control the duration of the algorithm.

Besides the complexity of the algorithm, one of the main limitations of the SemStim algorithm in its current form, is that SemStim does not take the semantics of the different edge types into account. All algorithms derived from spreading activation can assign different weights to different edge types. However SemStim currently uses a uniform weight β for all edges in the graph. The main benefit of using a uniform weight, is that SemStim can be as an unsupervised approach for any combination of domains, without (a) any pre-processing of the graph and (b) without first requiring to learn weights depending on the choice of target and source domain.

However, pre-processing of the graph using e.g. entropy measures as described by [Dehmer & Mowshowitz, 2011] are a promising direction for future research into improving the performance of the algorithm. The entry of a link type could be used to set the weight of that link type.

In addition, using machine learning approaches to learn the weight of different edge types, could be used in addition to pre-processing the graph. It might allow improving the performance of SemStim in general, and for specific domains. [Ostuni et al., 2013] can be seen as an example for integrating machine learning with a graph algorithm.

4.3.1. Contributions of this chapter

In this chapter we have formally described SemStim, our cross-domain recommendation algorithm. SemStim uses Linked Open Data from the DBpedia semantic network to find indirect connections between items from different domains. We have described how Linked Open Data provides indirect links between domains in Section 2.3 of Chapter 2.

In this chapter we have addressed research question (Q4):

Research question (Q4): Which recommendation algorithm can generate cross-domain recommendations, without any overlap between source and target domain and without any ratings for the target domain?

SemStim provides this capability, as it operates on DBpedia as a semantic network. This allows using a user profile with nodes from a source domain to generate recommendations from a different set of nodes as the target domain. SemStim then will use the semantic network to find indirect connections between the two sets of nodes. In addition, the non-linear activation model of SemStim enables the algorithm to return personalised results. We demonstrate the ability of SemStim to generate cross-domain recommendations in a cross-domain accuracy experiment as part of our Evaluation Chapter, in Section 5.4.

SemStim can be used as the recommendation algorithm of the personalisation component of the conceptual architecture for Linked Open Data recommender systems, which we introduced in Chapter 3. We will evaluate SemStim for both single-domain and cross-domain recommendation tasks in Chapter 5. In addition, we describe how we implemented and deployed SemStim in a real-world use case in Chapter 6.

Chapter 5.

Evaluation*

In this chapter, we present the results of evaluating our SemStim algorithm. SemStim has been designed to provide cross-domain recommendations using multi-source and domain-neutral user profiles. As this is a novel paradigm for recommender systems, we first need to show that SemStim is also able to generate single-domain recommendations.

This is followed by an evaluation of the cross-domain recommendations generated by SemStim. We aim to evaluate the diversity of the recommendations generated by SemStim, as diverse recommendations can lead to higher user satisfaction and engagement [Winoto & Tang, 2008].

Therefore, the objective of our evaluation is four-fold:

- The first objective, is to show that SemStim is a recommendation algorithm which can provide results with an accuracy that is comparable to established baselines *without* the use of any rating data from the target domain. We show this with an experiment in which we compare the accuracy of SemStim with both graph-based and non-graph-based algorithms. In this experiment we use MovieLens 100k rating data as the source and target domain.
- The second objective is to demonstrate the capability of SemStim to generate cross-domain recommendations, with different source and target domains. We show this with an experiment in which we use profiles with ratings in two different domains for this experiment. The preferences in one domain are used to generate recommendations in the second domain, and the ground truth is used to calculate precision, recall and F1-score. For this experiment we will use a data set with Amazon ratings from the domains of music, books, DVDs and videos.
- The third objective, is to show that SemStim can provide sets of recommendations which can be tuned to be *either* more diverse or more similar. We show this with an

*Section 5.6 of this chapter is based on previously published work:

1. Benjamin Heitmann, Conor Hayes, "SemStim at the Linked Open Data-enabled Recommender Systems 2014 challenge", In *Proceedings of the Extended Semantic Web Conference*, 2014.

experiment, in which we compare the diversity of the recommendations generated with SemStim with the diversity of recommendations generated by the algorithms used for comparison. For this experiment, we will use the MovieLens 100k rating data as target and source domain.

- The fourth objective is to investigate the relationship between accuracy and diversity of the recommendations generated by SemStim. While SemStim can provide both accurate and diverse recommendations, we want to evaluate if these two properties can be balanced. The diversity recommendation task at the Linked Open Data-enabled Recommender Systems challenge of the Extended Semantic Web Conference 2014, provided an opportunity for us to evaluate this trade-off between accuracy and diversity.

Contributions of this chapter: In this chapter we address research questions (Q1) and (Q4).

(Q1) asks how the cold-start problem for new users can be mitigated. We propose to use SemStim to mitigate this part of the cold-start problem. As SemStim can provide cross-domain recommendations, it is able to generate recommendations from a target domain if the user has no preferences in the target domain, as long as preferences from other domains are available for the user.

(Q4) asks for an algorithm that can generate cross-domain recommendations when there is no overlap between the users of two domains and no overlap between the items of the same two domains.

In Section 5.4, we describe our cross-domain accuracy experiment. It demonstrates how SemStim is able to provide cross-domain recommendations for domains without overlap. This in addition also shows how recommendations for a target domain can be generated from user preferences in a different source domain without any user preferences in the target domain. The results of the experiment show that SemStim can address the cold-start problem for new users.

Chapter outline: First, in Section 5.1, we describe the baseline algorithms to which we compare SemStim. As non-graph-based algorithms, we use two collaborative filtering implementations and a random selection worst case baseline. The collaborative filtering algorithms are k-nearest neighbour user-based collaborative filtering, and SVD++, which is a collaborative filtering algorithm based on matrix factorisation. As graph-based baselines we use *linked data semantic distance* (LDSD) [Passant, 2010b], and *set-based breadth first search* (SetBFS).

Then, in Section 5.2, we describe the data sets which we use for our different experiments. We provide the details of the MovieLens data set, the DBpedia data set, the Amazon multi-domain rating data set and the DBbook data set.

This is followed by descriptions of our experiments. For each experiment, we describe the experiment methodology and the results.

We describe our single-domain accuracy experiment on MovieLens data in Section 5.3. Then we describe our cross-domain accuracy experiment on Amazon multi-domain rating data in Section 5.4. This followed by the description of two experiments related to the diversity of the recommendations generated by SemStim. First we describe our experiment for evaluating the diversity of all algorithms on a single-domain in Section 5.5. For this experiment we use MovieLens 100k rating data as the source and target domain. The last experiment which we describe in Section 5.6, is our participation at the Linked Open Data-enabled Recommender Systems challenge at the Extended Semantic Web Conference 2014.

5.1. Baseline algorithms

In this section we describe the recommendation algorithms which we use in our evaluation for comparison with SemStim. We compare SemStim to five other recommendation algorithms. They come from the family of graph-based algorithms, which can use the topology of a network or graph for the recommendation task, and from "traditional" algorithms, which are not graph-based, as they use e.g. ratings to make recommendations.

From the group of non-graph-based recommendation algorithms, we use two Collaborative Filtering algorithms for our evaluation. These are k-nearest neighbor collaborative filtering (CFknn), and SVD++, which is a collaborative filtering algorithm based on matrix factorisation. In addition, we use random movie selection as a worst case non-graph-based baseline.

From the group of graph-based recommendation algorithms, we use Linked Data Semantic Distance (LDSD) [Passant, 2010b], as it makes use of a semantic network, and it is well known in the Semantic Web research community. In addition, we also use a graph-based baseline, which makes use of the network topology. We use set-based breadth first search (SetBFS) for this, which is a modified version of breadth first search.

We selected these two graph-based algorithms, as they fulfill the requirements for a cross-domain recommendation algorithm in our cross-domain personalisation framework, which we introduced in the Background chapter, in Section 2.2.3. These requirements are, that the algorithm must be able to perform the cross-domain recommendation task without any overlap between the source and target domain. In addition, the algorithm must not require any rating data for the target domain.

Both LDSD and SetBFS fulfill these requirements, as the do not use any rating data to make their recommendations, and they do not require any overlap between the domains, as they use DBpedia as a bridge between the source and target domain.

Depending on the type of experiment we used different algorithms for comparison, which is described in more detail as part of the respective experiments.

5.1.1. SVD++

SVD++ is state of the art collaborative filtering algorithm, which is based on matrix factorisation. It is first described by Koren [Koren, 2008]. We use the implementation of SVD++ which is provided by the GraphLab collaborative filtering toolkit[1]. GraphLab was originally developed at Carnegie Mellon University. We use SVD++ with 200 latent factors, as described by Cremonesi et al. [Cremonesi et al., 2010] at the start of their Section 4.

5.1.2. k-Nearest Neighbour Collaborative Filtering

In order to compare SemStim to a non-graph-based recommendation algorithm, we choose the family of Collaborative Filtering (CF) algorithms. In particular, we use a user-based k-nearest neighbour CF (CFknn) algorithm, as described by [Herlocker et al., 1999] and [Su & Khoshgoftaar, 2009].

CF is used to predict the rating of an item for a specific user. In its most basic form, a CF algorithm has two steps. As input, the algorithm requires a given user a, also called the active user, and an item i, as well as the user-item rating matrix R, which contains the rating vectors of all users. In the first step, the neighbourhood of the active user a is determined. This neighbourhood consists of the rating vectors of the most similar k users as determined by a similarity function. In the second step, the rating of an unrated item i by the active user a is predicted using the rating vectors in the neighbourhood of the active user.

Different methods for determining the neighbourhood and predicting an item rating are available, see [Su & Khoshgoftaar, 2009] for a list. In our implementation we use the vector cosine similarity for the similarity computation. For the rating prediction we use the weighted average of deviations from the neighbours mean rating of the item. Both methods are well understood and easy to implement.

The vector cosine similarity between users u and v is given by

$$cos(u, v) = \frac{u \bullet v}{\|u\| \cdot \|v\|} \tag{5.1}$$

where "$\bullet$" denotes the dot-product between two vectors, and "$\cdot$" denotes the multiplication of two scalars.

[1] http://docs.graphlab.org/toolkits.html

In order to predict a rating for an item, we use the following formula from [Su & Khoshgoftaar, 2009], to determine the weighted average of deviations from the mean item rating of i in the neighbourhood of the active user a:

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_u) \cdot w_{a,u}}{\sum_{u \in U} |w_{a,u}|} \tag{5.2}$$

$\bar{r}_a$ and $\bar{r}_u$ are the average ratings for the user a and user u on all other rated items, and $w_{a,u}$ is the similarity between user a and u. The summations are over all the users $u \in U$ who rated the item i.

5.1.3. Verification of our knn-CF implementation

We verified our implementation of CFknn, by replicating two experiments from the recommender systems literature. The goal of verifying our implementation is to show that it performs correctly in two different recommendation tasks, the rating prediction task and the top-n recommendation task.

The rating prediction task involves using each algorithm to predict the rating of a list of unrated items based on the known preferences of a given active user. The top-k recommendation task involves ranking a list L of unrated items for the active user. We go into more details about the differences between these two experiment protocols in Section 2.1.2.

The rating prediction task is important for the verification because it is very well understood in the recommender systems community. However, SemStim does not use rating data, as it is only able to perform the top-n recommendation task. This requires us to compare the performance of SemStim to CFknn using the top-N recommendation task. So we want to additionally show that our CF implementation performs the top-N recommendation task correctly. For the rating prediction task we replicated the experiment of [Herlocker et al., 1999]. For the top-N recommendation task we replicated the experiment of [Cremonesi et al., 2010].

First we verified the correctness of our CF implementation using the rating prediction experiment in [Herlocker et al., 1999]. Herlocker uses a historic snapshot of the MovieLens data from 1999, which consists of 122176 ratings and 1173 users, where each user has at least 20 ratings. 10% of the user profiles get randomly selected for the experiment. For each of these users, the profile is split into a test part and a training part. The test part contains a fixed number of 5 items, for which the rating in the original data is withheld. The remaining items go into the training part of the user profile. The recommendation algorithm then tries to predict the ratings of the 5 items in the test profile. The mean average error (MAE) between the predicated rating and the real rating is the main metric

used by Herlocker. The MAE results reported by Herlocker range from 0.72 to 0.78, depending on the specific details of the similarity and rating prediction functions used.

We replicated this experiment using the MovieLens 100k data set, as the exact historic MovieLens snapshot used in the paper is not publicly available. The MovieLens 100k data set contains 100000 ratings from 943 users. We used a neighbourhood size of 50. Our implementation of k-nearest neighbour CF had an MAE of 0.72 which is at the lower bound of the results reported by Herlocker. This shows that our implementation performs correctly with regards to the rating prediction task.

For verifying the correctness of our CF implementation for the top-N recommendation task, we replicated the experiment in [Cremonesi et al., 2010]. Cremonesi et al. suggest an experiment design, which allows evaluating the precision and recall of a recommender algorithm. We will use this experiment design to compare the accuracy of SemStim with the other graph algorithms and with the two CF implementations, which we describe in Section 5.3. In this experiment design, the training and test set are created from the full data set as follows: A sample of 1.4% of the data set is extracted, which is called the probe set P. All the 5-star ratings from the probe set form the test set T. The remaining ratings form the training set M.

For each item i in the test set T, the following sequence of steps is performed:

1. 1000 additional items which have not been rated by the active user are selected.
2. The ratings for item i and the remaining 1000 items are predicted.
3. A ranked list of all 1001 items is formed by sorting them according to their predicted rating in descending order. The rank of item i in this ranked list is denoted by p.
4. The list of top-N recommendations is formed by taking the first N items from the ranked list. If $p \leq N$ then the recommendation counts as a hit, otherwise it counts as a miss.

As there is only a single relevant item for each test case, modified versions of precision and recall can be used for the Cremonesi experiment:

$$\text{recall}(N) = \frac{\#\text{hits}}{|T|} \tag{5.3}$$

$$\text{precision}(N) = \frac{\#\text{hits}}{N \cdot |T|} = \frac{\text{recall}(N)}{N} \tag{5.4}$$

Cremonesi et al. use an unspecified version of the MovieLens data set, and they report a recall of 28% for $N = 20$ for their implementation of the k-nearest neighbourhood CF algorithm. We replicated this result using the MovieLens 1M data set, which contains

1000000 ratings from 6040 users. Otherwise we used the same experiment design. Our implementation of k-nearest neighbour CF achieved a recall of 24.4% for $N = 20$. This shows that our implementation of CFknn has performance for the top-N recommendation task which is comparable to published results.

5.1.4. Linked Data Semantic Distance

As a graph-based baseline for our experiments, we choose Linked Data Semantic Distance (LDSD). LDSD as a recommendation algorithm is well known in the Semantic Web research community. The algorithm was first described in [Passant, 2010b], at the AAAI Spring Symposium on "Linked Data meets Artificial Intelligence" in 2010. This was followed with the description of the dbrec recommender system in [Passant, 2010a], at the International Semantic Web Conference in 2010. In contrast to ontology-based recommender systems, such as the system presented by [Middleton et al., 2009], LDSD is a graph-algorithm for networks with typed edges, such as semantic networks. Ontology-based recommendation algorithms employ some form of reasoning for their recommendations, whereas LDSD and SemStim do not make use of the reasoning capabilities of Semantic Web technologies. This makes LDSD a suitable comparison algorithm for SemStim.

LDSD presents a distance metric between entities of a semantic graph. It is motivated by the fact that two resources are more related if they are the only ones sharing a particular property. Given a user profile with start nodes, LDSD returns the top-k most similar nodes, as determined by counting the number of direct and indirect links between each of the start nodes, and any 1st or 2nd degree neighbors from the target domain.

[Passant, 2010b] proposes multiple versions of LDSD, which are called the "direct distance", the "indirect distance" and the "combined distance".

The direct distance $LDSD_d(r_a, r_b)$ uses the function C_d. $C_d(p_i, r_a, r_b) = 1$, if $(r_a, p_i, r_b) \in G$. If the triple with subject r_a, the predicate p_i and the object r_b is part of the RDF graph G, then the functions is 1, else $C_d(p_i, r_a, r_b) = 0$. By extension, Passant defines $C_d(n, r_a, r_b) := \sum_{p_i} C_d(p_i, r_a, r_b)$ is the total number of direct and distinct links between r_a and r_b. In the notation of Passant, using n instead of p_i denotes a wildcard for the usage of the C_d function. The direct distance $LDSD_d(r_a, r_b)$ then is defined using C_d:

$$LDSD_d(r_a, r_b) = \frac{1}{1 + C_d(n, r_a, r_b) + C_d(n, r_b, r_a)} \tag{5.5}$$

The indirect distance $LDSD_i(r_a, r_b)$ uses the functions C_{io} and C_{ii}. C_{io} denotes the number of indirect and outgoing links between two nodes in G. $C_{io}(p_i, r_a, r_b) = 1$, if there is a node $m \in G$ for which $(r_a, p_i, m) \in G \wedge (r_b, p_i, m) \in G$. Else $C_{io}(p_i, r_a, r_b) = 0$. By extension, $C_{io}(n, r_a, r_b) := \sum_{p_i} C_{io}(p_i, r_a, r_b)$. Note that $C_{io}(p_i, r_a, r_b) = 1$ only if both r_a

and r_b have a link of type p_i to the same neighbour m, so that C_{io} is only applicable for indirect neighbours with a distance of 1 between each other.

Conversely, C_{ii} denotes the number of indirect and incoming links between two nodes in G. $C_{ii}(p_i, r_a, r_b) = 1$, if there is a node $m \in G$ for which $(m, p_i, r_a) \in G \wedge (m, p_i, r_b) \in G$. Else $C_{ii}(p_i, r_a, r_b) = 0$. By extension, $C_{ii}(n, r_a, r_b) := \sum_{p_i} C_{ii}(p_i, r_a, r_b)$.

The indirect distance $LDSD_i(r_a, r_b)$ is defined using C_{io} and C_{ii}:

$$LDSD_i(r_a, r_b) = \frac{1}{1 + C_{ii}(n, r_a, r_b) + C_{io}(n, r_a, r_b)} \tag{5.6}$$

The combined distance $LDSD_c(r_a, r_b)$ uses a combination of the direct and indirect distance:

$$LDSD_c(r_a, r_b) = \frac{1}{1 + C_d(n, r_a, r_b) + C_d(n, r_b, r_a) + C_{ii}(n, r_a, r_b) + C_{io}(n, r_a, r_b)} \tag{5.7}$$

For our experiments, we use the combined distance $LDSD_c(r_a, r_b)$ as a graph based recommendation baseline, as it combines the other two variations of LDSD, which allows it to explore a bigger subset of the graph. While $LDSD_d(r_a, r_b)$ only takes direct neighbours into account, and $LDSD_i(r_a, r_b)$ only takes indirect neighbours into account, $LDSD_c(r_a, r_b)$ takes both direct and indirect neighbours into account.

5.1.5. Random selection

In addition to comparing SemStim to both graph-based and non-graph-based baseline algorithms, we compared SemStim to a worst-case random selection algorithm. It selects up to N random items, which have not been rated by the active user.

5.1.6. Set-based breadth first search (SetBFS)

As the final comparison algorithm, we selected a baseline algorithm that takes the topology of the semantic network into account. Set-based breadth first search (SetBFS) is a modified version of breadth first search, which starts from the set of nodes in the user profile, instead of starting from a single node. It is an iterative algorithm. In each iteration, all the neighbours of already selected nodes are selected. The algorithm terminates, as soon as N or more nodes from the target domain have been found, which are not also in the user profile. A random selection of the found nodes are then recommended to the user, if the number of found nodes is bigger than N. In this way, SetBFS provides a baseline that is still based on the network topology and the start nodes in the user profile.

5.2. Data sets

In this section we describe the details of the different data sets which we use in our evaluation. First, in order to use SemStim to generate recommendations, both single-domain and cross-domain, we use a subset of DBpedia. For our single-domain recommendation experiments, we use data from the MovieLens project, as it has been extensively studied by the recommender systems research community. as it is easily available and very popular in the recommender systems research community. For our cross-domain recommendation experiments, a data set is required which contains user profiles that span multiple domains. We use the "Amazon product co-purchasing network metadata" as published by the Stanford SNAP project [Leskovec et al., 2007] for our cross-domain recommendation experiments. Finally, in order to participate at the Linked Open Data-enabled recommender systems challenge, we used the DBbook data set, as provided by the organisers of the challenge.

5.2.1. MovieLens data set

The MovieLens project [Miller et al., 2003] has released multiple data sets. The data sets contain the user preferences of users of the MovieLens movie recommendation web site. Together these data sets might be the most frequently studied data sets in the recommender system community, besides the Netflix data set. However, while the Netflix data set is not officially available any more, the MovieLens data sets are available on the web site[2] of the project.

We used the MovieLens 100k data set for our single-domain recommendation experiments. The data set contains 100,000 ratings between 1 and 5, from 943 users on 1,682 movies, where each user has rated at least 20 movies.

In order to use the MovieLens data set with SemStim, the movies need to be linked to their equivalent resources on DBpedia. For each movie in the MovieLens data, the title, the release data, the genre and the Internet Movie Database (IMDb) URL are known. However, there is no source of linkage data between IMDb to DBpedia, so the availability of the IMDb URL can not be used for linking from MovieLens to DBpedia. Instead we use the OpenRefine[3] toolkit with the RDF extension[4] to generate the linkage data for the movies. OpenRefine provides a visual interface for importing, exploring and inspecting a data set. It also allows for linking one non-RDF data set to an RDF data set, using simple heuristics such as fuzzy string matching. OpenRefine allows selecting the linkage results with the lowest confidence, and manually changing them, so we manually fixed the

[2] http://grouplens.org/datasets/movielens/
[3] http://openrefine.org/
[4] http://refine.deri.ie/

20% of links with the lowest confidence. This results in a 98% coverage for the MovieLens movies with only 37 missing movies.

5.2.2. DBpedia data set

In order to generate recommendations with SemStim we use the RDF graph that is provided by DBpedia. The DBpedia [Lehmann et al., 2014] community project extracts structured, multilingual knowledge from Wikipedia and makes it freely available on the Web using Semantic Web and Linked Data technologies. The project extracts knowledge from 111 different language editions of Wikipedia. The largest DBpedia knowledge base which is extracted from the English edition of Wikipedia consists of over 400 million facts that describe 3.7 million things. The DBpedia project maps Wikipedia infoboxes from 27 different language editions to a single shared ontology consisting of 320 classes and 1,650 properties. The mappings are created via a world-wide crowd-sourcing effort and enable knowledge from the different Wikipedia editions to be combined. The project publishes releases of all DBpedia knowledge bases for download.

We use a subset of DBpedia 3.8 with 67 million edges and 11 million vertices. In particular, we use the following files which are part of DBpedia 3.8:

- All ontology infobox types, which are the `rdf:type` triples from the ontology-based extraction process. This data set only uses the DBpedia ontology. (13,225,167 triples, 1.8 GB size after unpacking, file name: `instance_types_en.nt`)
- ontology infobox properties, which is high-quality data extracted from Wikipedia infoboxes using the ontology-based extraction. (20.516.861 triples, 2.6 GB size after unpacking, file name: `mappingbased_properties_en.nt`)
- article categories: Links from DBpedia concepts to Wikipedia categories. (15,115,486 triples, 2.1 GB size after unpacking, file name: `article_categories_en.nt`)
- disambiguation links, which are links extracted from Wikipedia disambiguation pages. (1,157,484 triples, 164.7 MB size after unpacking, file name: `disambiguations_en.nt`)
- transitive redirects, are redirection links from Wikipedia in which multiple redirects have been resolved and redirect cycles have been removed. (5,521,268 triples, 807.5 MB size after unpacking, file name: `redirects_transitive_en.nt`)
- SKOS categories: Information about which concepts are categories and how categories are related to each other (e.g. broader or narrower) using the SKOS vocabulary. SKOS is the Simple Knowledge Organization System, an W3C standard designed for representation of thesauri and other types of structured controlled vocabularies. (3,458,049 triples, 564 MB size after unpacking, file name: `skos_categories_en.nt`)

- Links to the YAGO[5] ontology, which provide an additional way to classify DBpedia concepts, besides the DBpedia ontology. (8,171,197 triples, 1.2 GB size. This contains only the `rdf:type` triples from the file `yago_links.nt`)

We choose this subset, because SemStim as a graph algorithm requires as much available data about edges between vertices on DBpedia as possible, and this subset contains all the available edge data. Conversely we did not use any literals from DBpedia, as they are not used by SemStim. The data contains 72,000 movies, 26,000 books, 159,000 music records and 572,000 travel destinations.

5.2.3. Amazon multi-domain rating data set

For our cross-domain experiments, we need a data set with ground truth from multiple domains. The Stanford Network Analysis Project (SNAP)[6] provides a large collection of openly available network datasets. One of these data sets is the "Amazon product co-purchasing network metadata" set[7], which was collected in 2007 for research on the topic of viral marketing in social networks [Leskovec et al., 2007].

This data set contains 5,509,603 reviews of 548,552 products on Amazon in the four domains of "books", "music", "DVDs" and VHS "videos". These domains refer to the classification of items in the internal Amazon catalogue. The data contains ratings for 393,561 books, 103,144 music CDs, 19,828 DVDs and 26,132 videos. The total number of users is 1,389,641. Each rating is between 1 and 5 stars.

We link the Amazon data to DBpedia in two steps: first we pre-processed the data, then we create the links from the Amazon items to DBpedia using fuzzy string matching. The pre-processing of the data set serves the purpose of identifying the Amazon items which need to be linked to DBpedia in the first place. We remove items from the data set, based on four criteria:

1. We remove any items from the original data set which are marked as "discontinued", as these items have no ratings. These discontinued items are probably only included in the original data to make it consistent.

2. We remove all 1, 2 and 3 star ratings from the remaining data, as we are interested in positive ratings (4 to 5 stars). We can remove neutral and negative ratings, as we only use graph-based algorithms in our cross-domain experiment. Section 5.4.2 explains why both collaborative filtering implementations are unsuitable for the cross-domain experiment.

[5] http://www.mpi-inf.mpg.de/yago-naga/yago/
[6] http://snap.stanford.edu/
[7] http://snap.stanford.edu/data/amazon-meta.html

3. We remove the users (and their ratings) who have rated "too many" items in at least one domain, as they will skew the results of the experiment. In particular, we remove every user whose number of ratings in one domain is bigger then the 98-th percentile in the rating distribution of that domain. There is a very high likelihood that these outliers have automatically generated these ratings with malicious intent, in order to spam Amazon. The number of users removed in this way is 276 out of 1,389,641, which is less then 0.01%. These users usually have several thousand ratings in one domain.

4. We remove all users, who have exclusively rated items in one domain. As we will use the data for a cross-domain recommendation experiment, we are only interested in the population of users who have rated items in two or more of the four domains.

After pre-processing the data, 336,244 items remain, which need to be linked to DBpedia. That is 61.3% of all items in the original data set.

Then we create the links between the remaining Amazon items and DBpedia. We use Jena Fuseki[8] to provide a SPARQL end-point with full-text search capabilities for our DBpedia subset. Fuseki uses Apache Lucene[9] to provide the full-text search. We used Jena Fuseki to generate the Amazon linkage data instead of using OpenRefine, as we ran into scalability issues when attempting to use OpenRefine to link such a large number of items.

We generate the linkage data in two steps:

1. All characters which are not alpha-numeric, are removed from the Amazon title of an item.

2. Then we use the title to perform a full-text search on our DBpedia subset. We additionally provide an `rdf:type` for the search, depending on the Amazon domain of the item. However, if this search is empty, we fall back on the same search without any type information.

3. The top-5 results of the search are then used to link the respective item to 5 concepts on DBpedia.

Using this process we are able to link 99.7% of 336244 Amazon items. Only 971 items did not have any search results on DBPedia.

5.2.4. DBbook data set

In order to participate at the Linked Open Data-enabled recommender systems challenge, we used the DBbook data set, as provided by the organisers of the challenge. In particular,

[8] https://jena.apache.org/documentation/serving_data/
[9] https://lucene.apache.org/

as we only participated in the diversity task of the challenge, we used the training data in the `DBbook_train_binary.zip` file. There was no test data provided for the diversity task. We will discuss the details of the diversity task and the other two recommendation tasks in section 5.6.

The DBbook data set is has been generated by scraping LibraryThing[10]. The data contains 72,372 ratings by 6,181 users for 6,733 items. The ratings are in binary scale, where 1 indicates that the item is relevant for a user, and 0 means the item is not relevant. The linkage data for the items in DBbook to DBpedia is provided by the organisers.

5.3. Single-domain Accuracy Experiment

In our first experiment, we determine if SemStim is able to produce recommendations with an accuracy that is comparable or better than that of the baseline algorithms for the single-domain recommendation task. Alternatively this task might be called the "intra-domain recommendation task".

In this section, we will first describe the experiment protocol and the metrics of our single-domain accuracy experiment. Then we present our results and discuss the threats to the validity of our experiment.

5.3.1. Experiment protocol

In order to perform this single-domain accuracy experiment, we use the rating data from MovieLens, as described in 5.2.1.

We will focus on using the top-k recommendation task in our single-domain accuracy experiment instead of the rating prediction task, as the top-K recommendation task fits the intended use case of cross-domain personalisation in social networking ecosystems. We explain the difference between the top-k recommendation task and the rating prediction task in Chapter 2 in Section 2.1.2.

We have adapted our experiment protocol from [Cremonesi et al., 2010], who propose an experiment protocol that allows using accuracy metrics such as precision and recall for evaluating rating predictions. The main idea is to form a test set for each user by random sampling of the most positive ratings of the user, and then mixing them with a large number of random unrated movies. In this test profile, the originally highly ranked items are labelled as belonging to the "positive" class, while the random unrated movies are labelled as belonging to the "negative" class. The remaining part of the original user profile forms the training profile.

[10] http://sisinflab.poliba.it/semanticweb/lod/recsys/datasets/

Labeling the items in the test profile allows evaluating the performance of each recommendation algorithm in the same way as a binary classifier. Each recommendation algorithm ranks all of the items in the test profile. The parameter of k determines the cut-off point for the ranked list of recommendations. The top-k items in the ranked list of recommendations are then used to determine the number of "true positives". E.g. when evaluating the top-15 recommendations for a specific user and algorithm, if the first 15 items which are recommended include 3 items labelled as "positive", then the algorithm has recommended 3 "true positives" and 12 "false positives".

Please note, that the classification labels for the items in the test profile are always withheld from the recommendation algorithms.

For our specific experiment, we generate the training and test set from the full MovieLens 100k rating data set as follows: First we sample 10% of each MovieLens user profile into a probe set P. The remaining 90% of the data set form the training set M. We then construct the test set T as follows:

1. We take all highly rated items (all 4 and 5 star ratings) from the probe set P of a user, labeled as belonging to the positive class. The mean number of highly rated items per user is 6.48.

2. Then we add random, unrated items for each user, labeled as belonging to the negative class, so that each user profile in P contains 200 items. The resulting user profiles contain 96% items from the negative class on average. Cremonesi suggests a sample size of 1.4% for the probe set, however this resulted in just 2 highly rated items per user profile on average, so we increased the sample size to 10%, which results in 6.48 highly rated items per user on average.

We then generate a ranked list of recommendations for each of the different algorithms.

It is important to note that all algorithms have access to the same training data. However, while CF uses positive *and* negative ratings to find the neighbourhood of a user, the graph algorithms are each essentially implementing a similarity based approach to find similar items. Therefore the graph algorithms only use positive preferences to make recommendations.

We generate a ranked list of recommendations for each of the different algorithms as follows:

SemStim: The DBpedia URIs of all positively rated movies (4 and 5 stars) in the training profile M of the active user are used as the set P of start nodes for SemStim. The target target domain D is the set of DBpedia URIs for all movies in the test profile T of the active user. For the reported results, we used the following constants: activation threshold: $\tau = 0.4$; maximum number of waves: $w_{max} = 6$; maximum number of phases: $\rho_{max} = 2$; default weight of predicates: $\beta = 1.0$; initial node output before applying modifiers: $\alpha = 4.0$. Other configurations either performed

worse or were not able to reach the required number of target activations. We set $\theta = 20$ as the required number of activated nodes from the target domain. The target domain only includes items from the test profile of the active user. When the algorithm terminates, we rank the activated nodes from the test profile by their activation value in descending order and return the ranked list.

Linked Data Semantic Distance (LDSD): Given the training set M and the test set T for the active user, we determine the minimum LDSD distance for all positively rated items from the test set to any item in the training set: $\forall t_j \in T: \ LDSD(t_j) = \min_{m_i \in M} LDSD_c(m_i, t_j)$. We then rank all the items from the test set by their smallest distance to the training set, in descending order, and return this ranked list.

Set-based Breadth First Search (SetBFS): For each user, SetBFS starts from the positively rated items in the training set M of the active user, and terminates when 20 nodes from the test set T have been found in the DBpedia network. The nodes are then ranked in the same order in which SetBFS found them in the network, and the ranked list is returned.

SVD++ and Collaborative Filtering (CFknn): First a model is generated for both algorithms, by using all ratings in the training profiles of all users in the data set. Then for each user, ratings are predicted for each item in the test profile of the active user. The test profile items are then ranked in descending order by their predicted rating, and one ranked list for each of both algorithms is returned.

Random selection (Random): The Random algorithm returns 20 random items from the test set of the active user.

We determine *precision*, *recall* and *F1-score* for each algorithm and each user, by determining the number of true positives in each set of recommendations using the withheld classification labels. #true-positives@k refers to the number of items labelled as belonging to the "positive" class, among the first k items of the ranked list of recommendations. #all-positive-items refers to the total number of items in the test profile of the active user, which belong to the "positive" class.

$$\text{precision@k} = \frac{\#\text{true-positives@k}}{k} \tag{5.8}$$

$$\text{recall@k} = \frac{\#\text{true-positives@k}}{\#\text{true-positives-in-user-test-profile}} \tag{5.9}$$

$$\text{F1-score@k} = 2\frac{\#\text{precision@k} \cdot \#\text{recall@k}}{\#\text{precision@k} + \#\text{recall@k}} \tag{5.10}$$

5.3.2. Results

Figures 5.1, 5.2 and 5.3 show the results for the single-domain accuracy experiment using MovieLens as both source and target domain. The error bars in the graphs show the standard error.

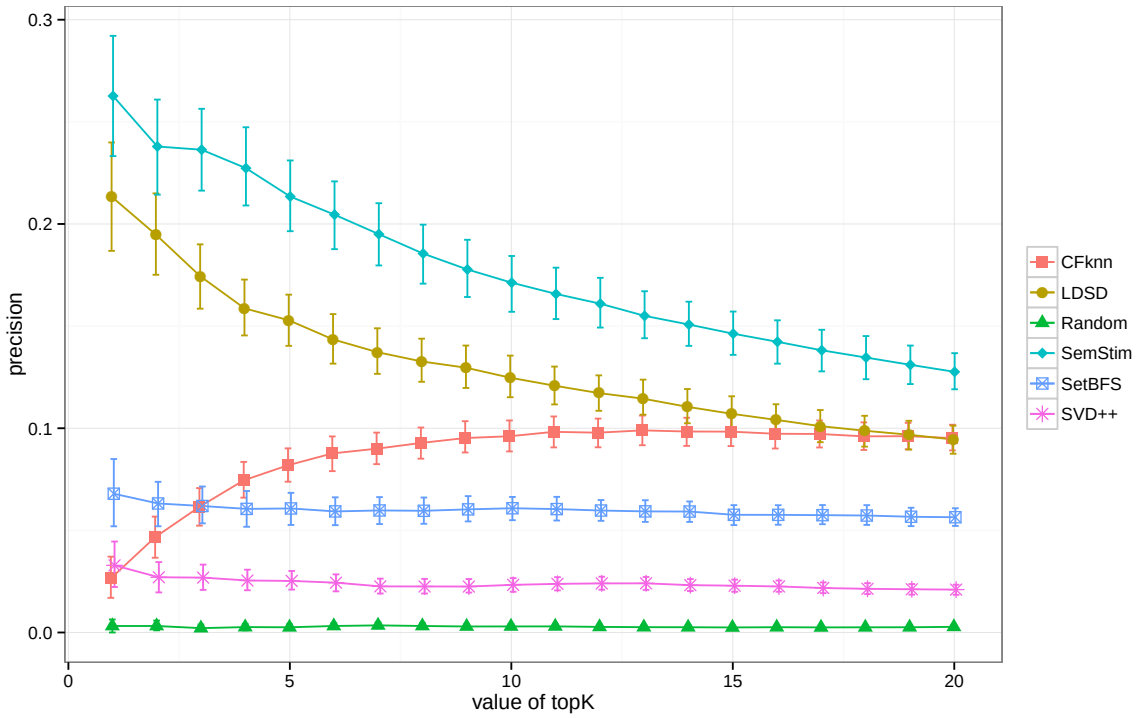

Figure 5.1.: Mean precision results for single domain top-k recommendations, with MovieLens as source and target domain. Error bars show the standard error.

Figure 5.1 shows the mean precision results for the top-k recommendation task. SemStim dominates the precision performance of the other algorithms, with precision values ranging from 26.3% for $k = 1$ to 12.8% for $k = 20$. The next-best performing algorithm is LDSD, with a precision around 4% worse than SemStim for all values of k. Both LDSD and SemStim show the same trend of diminishing precision with growing k. CFknn shows a different trend with precision increasing from 2.6% for $k = 1$ to 9.9% for $k = 13$, where it reaches a plateau approaching the same precision as LDSD. Finally, the random baseline, SVD++ and SetBFS have a constant precision which is independent from k. The precision of the random baseline is constantly around 0.3%, SVD++ achieves a precision of 2.4%, and SetBFS achieves a precision of 5.9%. The differences between SemStim and the other algorithms are statistically significant, as the error bars do not overlap.

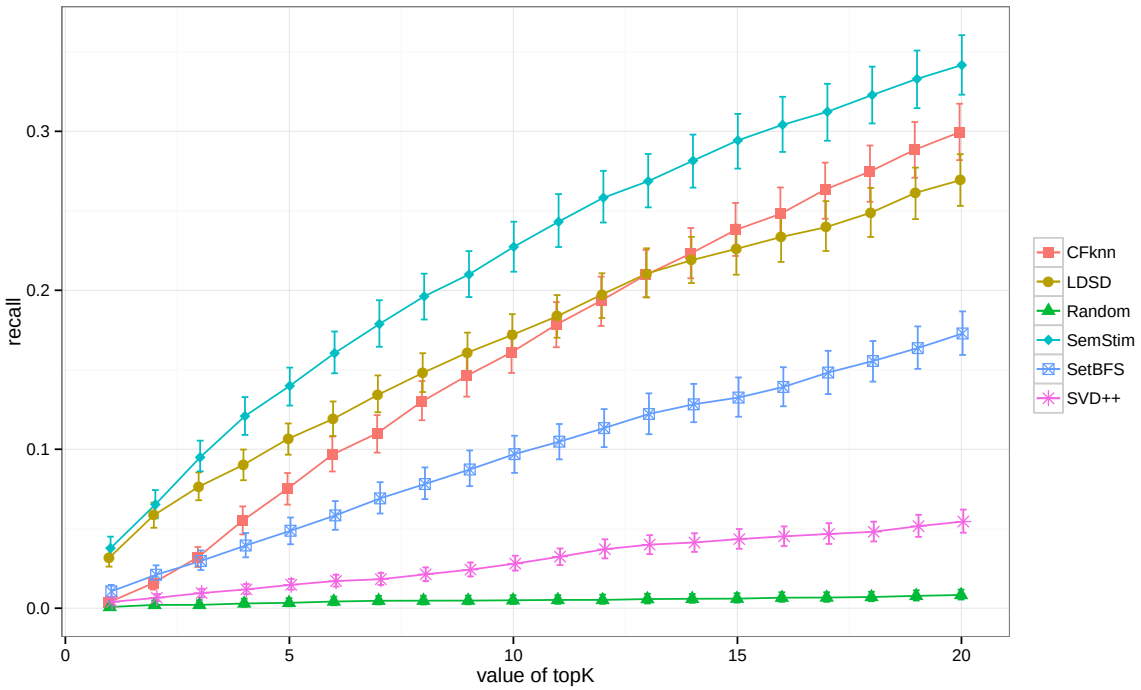

Figure 5.2.: Mean recall results for single domain top-k recommendations, with MovieLens as source and target domain. Error bars show the standard error.

Figure 5.2 shows the mean recall results for the top-k recommendation task. SemStim dominates the recall performance of the other algorithms. All algorithms show the same trend of noticeably increasing recall for increasing k. SemStim has recall values ranging from 3.4% for $k = 1$ to 34.2% for $k = 20$. LDSD and CFknn show very similar results for recall, however both are outperformed by SemStim by up to 6.0%. Of the two "random" baselines, SetBFS performs a lot better than random selection, with SetBFS at 17.2% for $k = 20$ and Random at 0.8% for $k = 20$. SVD++ is the worst performing non-random algorithm with 5.4% for $k = 20$. The differences between SemStim and the other algorithms are statistically significant, as the error bars do not overlap for $k > 3$.

Figure 5.3 shows the mean F1-score results for the top-k recommendation task. SemStim dominates the F1 performance of the other algorithms. The next-best performing algorithm is LDSD, followed by CFknn. The trend of the SemStim F1 values describes a curve starting at 6.0% for $k = 1$, peaking at 17.4% for $k = 12$ and decreasing down again to 16.7% at $k = 20$. All other algorithms show a similar trend, however they are outperformed by SemStim by margin larger than 6.0%. LDSD is the next best performing algorithm, which peaks at 13.0% for $k = 13$. Next, CFknn peaks at 12.9% for $k = 20$. SetBFS peaks at 7.5% for $k = 20$. SVD++ is outperformed by algorithms except the Random baseline. SVD++ peaks at 2.7% for $k = 20$, while the Random algorithm only reaches 0.4% for $k = 20$. The differences between SemStim and the other algorithms are statistically significant, as the error bars do not overlap for $k > 5$.

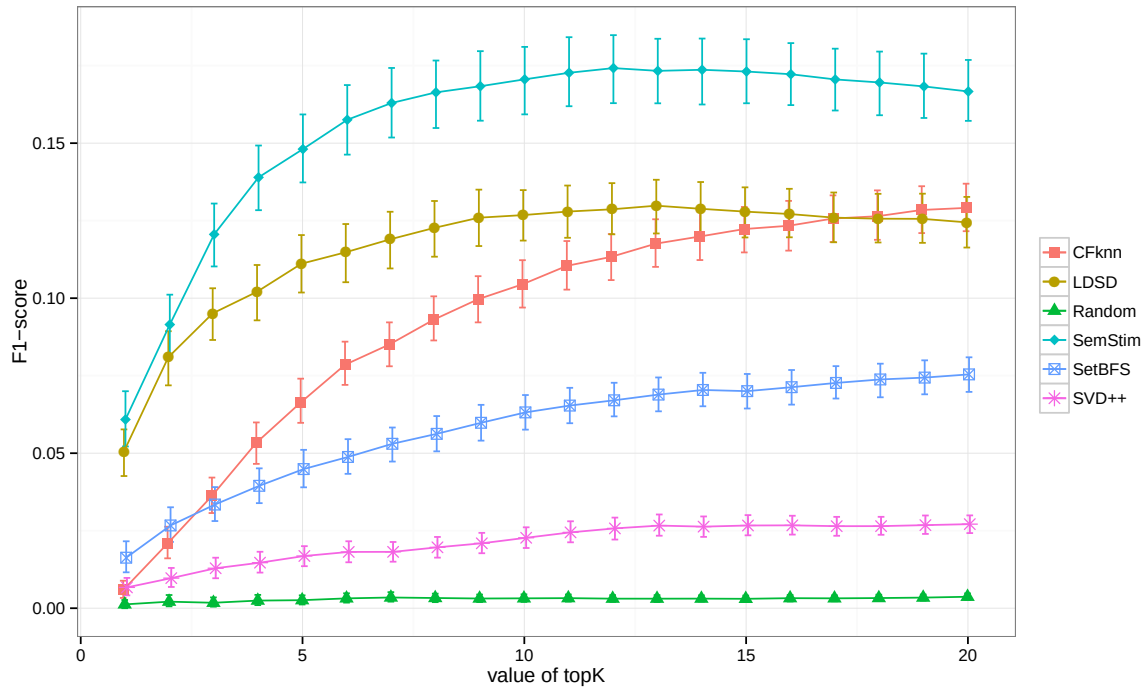

Figure 5.3.: Mean F1-score results for single domain top-k recommendations, with MovieLens as source and target domain. Error bars show the standard error.

5.3.3. Discussion

The results show that in our single-domain accuracy experiment on MovieLens data, SemStim can provide top-K recommendations, which are more accurate than the results of the baseline algorithms, including collaborative filtering. SemStim dominates the performance results for all three metrics of precision, recall and F1-score.

However, due to the prevalence of experiments based on the rating prediction task in the recommender systems community, we will now address the main threats to the validity of our experiment. These are (1) the validity of the experiment protocol used in our top-k recommendation experiment, (2) the validity of our CFknn implementation, (3) the plausibility of our results and (4) the significance of our results.

Experiment protocol

First, we discuss the validity of our experiment protocol, which is described in detail in Section 5.4.4. This experiment protocol is based on the protocol described by [Cremonesi et al., 2010], which is the most cited paper in the recommender systems community on the topic of top-k recommendations at the time of writing. We adhere to the proposed protocol except for the following changes, which are required by the properties of the MovieLens 100k data set:

1. We sample 10% of each user profile into the probe set P instead of 1.4%. This is required in order to have a sufficient number of highly rated items in each probe set.
2. We use all 4 and 5 star ratings from the probe set to form the test profile, instead of using only 5 star ratings. Again, this is required in order to have a sufficient number of highly rated items in the test profile of each user (6.5 instead of 2 on average).

Therefore our experiment protocol is based on a protocol which is trusted by the recommender systems community.

Validity of CFknn implementation

Secondly, we discuss the validity of our CFknn implementation.

The implementation of SVD++, which we use in the experiment, is provided by the GraphLab collaborative filtering toolkit[11]. This is one of the most used collaborative filtering libraries in the recommender systems research library. GraphLab was originally developed at Carnegie Mellon University. We use SVD++ with 200 latent factors as described in published research by [Cremonesi et al., 2010] at the start of their Section 4.

We have verified our implementation of user-based CFknn (cf. Section 5.1.3) by replicating published results for RMSE performance of CFknn in [Herlocker et al., 1999], as well as by replicating published results for the top-K performance of CFknn in [Cremonesi et al., 2010]. This shows that our CFknn implementation performs correctly. More generally speaking, the findings of our experiment agree with the findings of the experiment performed by Cremonesi et al. in [Cremonesi et al., 2010]. The authors suggest that "algorithms optimized for minimizing RMSE do not necessarily perfom as expected in terms of the top-N recommendation task".

Plausibility of results

Thirdly, we discuss the plausibility of our results.

Both collaborative filtering algorithms are out-performed by SemStim. This agrees with the findings of the experiments performed by [Cremonesi et al., 2010]. The authors suggest that algorithms, such as CFnn and SVD++, which are optimized for minimizing RMSE do not necessarily perform as expected in terms of the top-N recommendation task.

Further, the results show that algorithms which operate on shared principles have similar performance trends. SemStim and LDSD are both deterministic, graph-based

[11] http://docs.graphlab.org/toolkits.html

algorithms. They achieve the best results for all three metrics, with SemStim followed by LDSD. At the low end of the performance scale, we find the two non-deterministic and random algorithms SetBFS and Random selection. SetBFS has the role of a worst-case graph-based baseline, while Random has the role of a worst case non-graph baseline. Both algorithms consistently perform worst, with SetBFS always outperforming Random. Finally, CFknn is the only algorithm which is deterministic and not graph-based. For this experiment design, CFknn performs worse than SemStim, but better than both worst-case baselines. The performance of LDSD and CFknn is very close, LDSD having a better recall for $k \leq 10$ and CFknn having a better recall for $k > 10$, and with LDSD performing slightly better than CFknn for all other metrics.

As each of the algorithms has a performance which is similar to the other algorithm in the same algorithm class (graph-based, non-graph-based and worst-case baseline), this shows that our results are plausible.

Significance of results

Finally, we discuss the statistical significance of our results. The graphs for precision, recall and F1-score all show error bars for the standard error. For values of $k > 5$, neither the error bars of the SemStim curve, nor the means of the SemStim curve are overlapping with any other error bars or means of other algorithms. Therefore the differences between SemStim and all other algorithms for precision, recall and F1-score are statistically significant.

In summary, the results of our experiment are valid, as we have used a trusted experiment design. Further, we have verified our implementation of the CFknn algorithm. In addition, the results of our experiment are plausible and statistically significant.

5.4. Cross-domain accuracy experiment

After evaluating the accuracy of SemStim for the single-domain recommendation task, we show that we can use SemStim to generate cross-domain recommendations for different source and target domains.

In order to perform this cross-domain experiment, we require a data set with ground truth from multiple domains. However, most existing data sets which could provide cross-domain user profiles are not openly available.

One data set which provides cross-domain user profiles, is the "Amazon product co-purchasing network metadata" set[12]. It provides ratings from the four domains of books, music, DVDs and videos.

[12] http://snap.stanford.edu/data/amazon-meta.html

However, this data set was not collected for the purpose of cross-domain personalisation. It was collected in 2007 for research on the topic of viral marketing in social networks [Leskovec et al., 2007]. It is made available by the Stanford Network Analysis Project (SNAP)[13]. We describe the details of this data set and how we pre-processed it, in Section 5.2.3. We will refer to this data set from now on as the "Amazon Snap" data set.

As the "Amazon Snap" data set has not been collected for the purpose of cross-domain personalisation, the majority of the user profiles in the Amazon Snap data set are not cross-domain user profiles. While the data set does contain ratings from the four domains of books, music, DVDs and videos, most users have only ratings in one of these domains. As we will show, this leads to a a very high sparsity of ratings for users that have ratings in at least two of the four domains.

For our experiment, we first determine the population of all users who rated at least one item in any combination of two domains. From these populations, we then select the population with the smallest amount of rating sparsity.

Then, before the cross-domain accuracy experiment, we first perform a single-domain accuracy experiment with the ratings from that user population. The goal of that experiment is to determine if the high sparsity of the ratings might exclude any algorithm from the cross-domain experiment.

Then we use the algorithms, which are suitable to the high sparsity of the ratings, in order to perform a cross-domain recommendation experiment with the two domains from the ratings of the same population.

In this section we first describe the sparsity of the Amazon SNAP data set. Then we describe the experiment protocol and the results of the single-domain accuracy experiment on the Amazon SNAP data. This is followed by our description of the experiment protocol and results of the cross-domain experiment on the Amazon SNAP data.

5.4.1. Sparsity analysis of Amazon SNAP cross-domain profiles

We use the SNAP Amazon data set, as it contains ratings for the four domains of books, music, DVDs and VHS videos. For our experiment, we require user profiles with preferences from at least two of the four domains. In addition, we only use profiles with at least 20 ratings for at least two domains, as this provides 20 preferences for both train and test profiles.

However, the majority of the user profiles in the Amazon Snap data set are not cross-domain user profiles. The total number of items in the data set is 542,684, of which 39,3561 are books, 103,144 are music, 19,828 are DVDs, and 26,132 are videos. On the

[13] http://snap.stanford.edu/

other hand, the total number of users who rated at least one item in any domain is 1,389,641. Of these users, 861,270 rated books, 429,878 rated music, 207,698 rated DVDs, and 210,724 rated videos.

However only 295,408 (21%) of the 1,389,641 user profiles contain one or more ratings in at least two domains. This leaves 1,094,233 (79%) user profiles which do not contain any cross-domain preferences, and as such do not provide any ground truth for this cross-domain experiment.

In order to perform recommendations, user profiles of a certain size are required. As the user profiles in the MovieLens 100k rating data set contain at least 20 ratings each, we decided to also use 20 ratings as the lower bond to select which pair of domains to use. When requiring at least 20 ratings in any pair of two domains, we are left with the following population sizes:

- music and videos: 803 users.
- music and DVDs: 845 users.
- books and music: 1061 users.
- books and videos: 1255 users.
- books and DVDs: 1271 users.
- DVDs and videos: 2653 users.

Each of these populations is below 1% of the total population size of 1,389,641 users.

Due to the small size of these populations, we checked for the sparsity of the ratings of each population.

We use a definition of sparsity, which is also used e.g. by the MyMediaLite [Gantner et al., 2011] recommender system library, however we adapt it for two domains:

$$\text{sparsity}(Domain_1, Domain_2) = 1 - \frac{|ratings(Users_1 \cap Users_2)|}{|Users_1 \cap Users_2| \cdot (|Items_1| + |Items_2|)} \quad (5.11)$$

The combined sparsity of ratings for the populations of users with 20 ratings in any pair of two domains, is as follows:

- music and videos: 99.9889%.
- music and DVDs: 99.9883%.
- books and videos: 99.9948%.
- books and music: 99.9962%.

- books and DVDs: 99.9946%.
- DVDs and videos: 99.9855%.

All the sparsity values are very high, which shows the difficulty of using this data set for cross-domain recommendation. For comparison, the MovieLens 100k rating data has a sparsity of 93.6953%.

The pairing of music and DVDs has the lowest sparsity of 99.9883%. So we choose the population of users with 20 ratings for both music and DVDs for our cross-domain experiment.

While the pairing of DVDs and videos has the lowest sparsity of 99.9855% and the biggest population size of 2653, we can not consider this pairing for our cross-domain experiment, as most movies are available in more then one format. The most likely explanation for the fact that DVDs and videos are considered as separate domains in the Amazon SNAP data, is that this was required for reasons of inventory keeping and accounting. The Amazon SNAP data set was created in 2007, at which time e.g. DBpedia did not distinguish between movie releases in DVD form and other formats.

Due to the results of our sparsity analysis, we will use the pairing of the two domains music and DVDs for our cross-domain experiment, as it has the lowest sparsity and the highest population of users.

5.4.2. Sparsity test of all algorithms using Amazon SNAP data

Due to the small size of the "best" user population, and the high rating sparsity of that population, we firstly decided to perform a single-domain recommendation experiment on the selected population. This allows us to determine if the sparsity of ratings is having an adverse effect on the two CF algorithms.

Experiment protocol

For this sparsity test, we use exactly the same experiment protocol as in the single-domain accuracy experiment in Section 5.3. The only difference is that instead of using ratings from the MovieLens 100k rating data to generate the train and test profile of each user, we use the DVD rating data from the population of users with 20 ratings for both music and DVDs.

For this experiment, we want to use only ratings for one domain. However, we want this to be one of the two domains which we will use in our cross-domain experiment. The DVD domain by itself has 207,698 users, 19,828 items and 603,407 ratings, resulting in a

sparsity of 99.98535%. The music domain by itself has 429,878 users, 103,144 items and 1,084,733 ratings, resulting in a sparsity of 99.99755%.

Therefore we choose the DVD ratings of this population for this sparsity test, instead of the music ratings, as the DVD ratings have the lower sparsity.

We use all algorithms in the experiment. For SemStim we show the results for using three values (0.2, 0.3 and 0.4) for the activation threshold τ.

Results

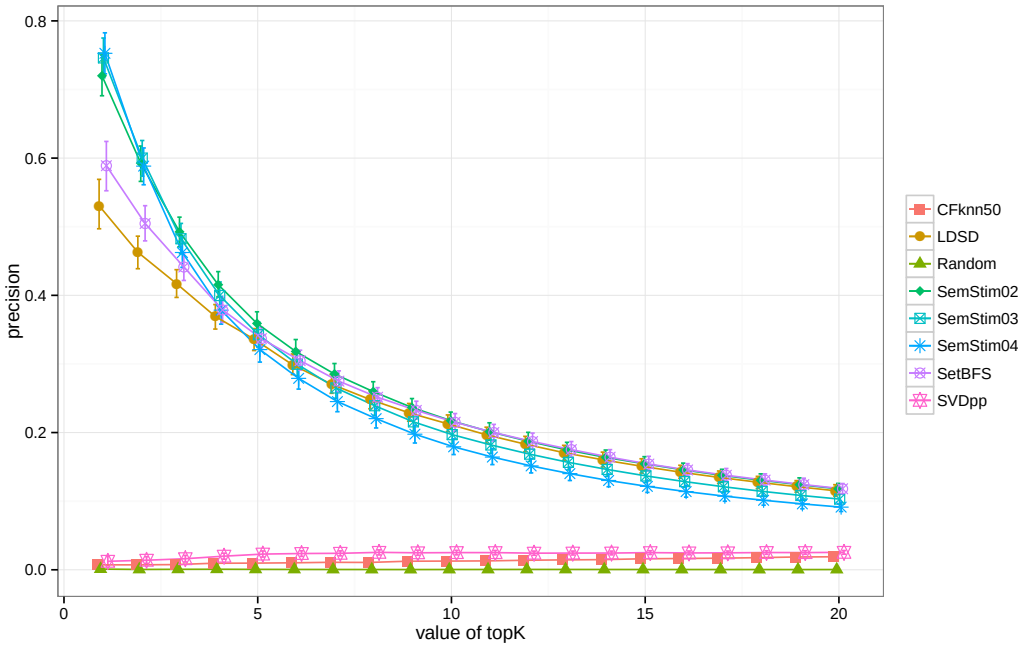

Figure 5.4.: Mean precision results for single-domain top-k recommendations, with DVD ratings as source and target domain. Error bars show the standard error.

Figures 5.4, 5.5 and 5.6 show the results of the single-domain sparsity test.

Each algorithm shows a similar trend among all three metrics of precision, recall and F1-score. The graph algorithms have the best performance. The best performing graph algorithms are SetBFS and SemStim with $\tau = 0.2$. The differences between SetBFS and SemStim02 are not statistically significant. The two SemStim variations with $\tau > 0.2$ perform worse then SetBFS and LDSD. However, the non-graph-based algorithms Random, CFknn and SVD++ have the worst performance of the experiment. Both CF algorithms have results which are roughly an order of magnitude worse than the graph algorithms.

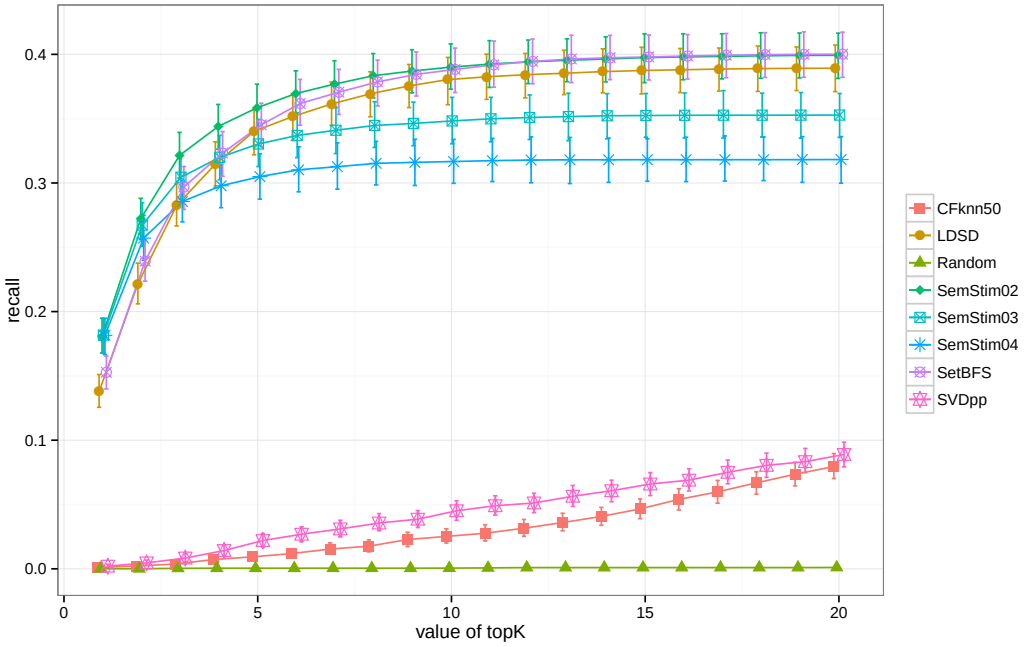

Figure 5.5.: Mean recall results for single-domain top-k recommendations, with DVD ratings as source and target domain. Error bars show the standard error.

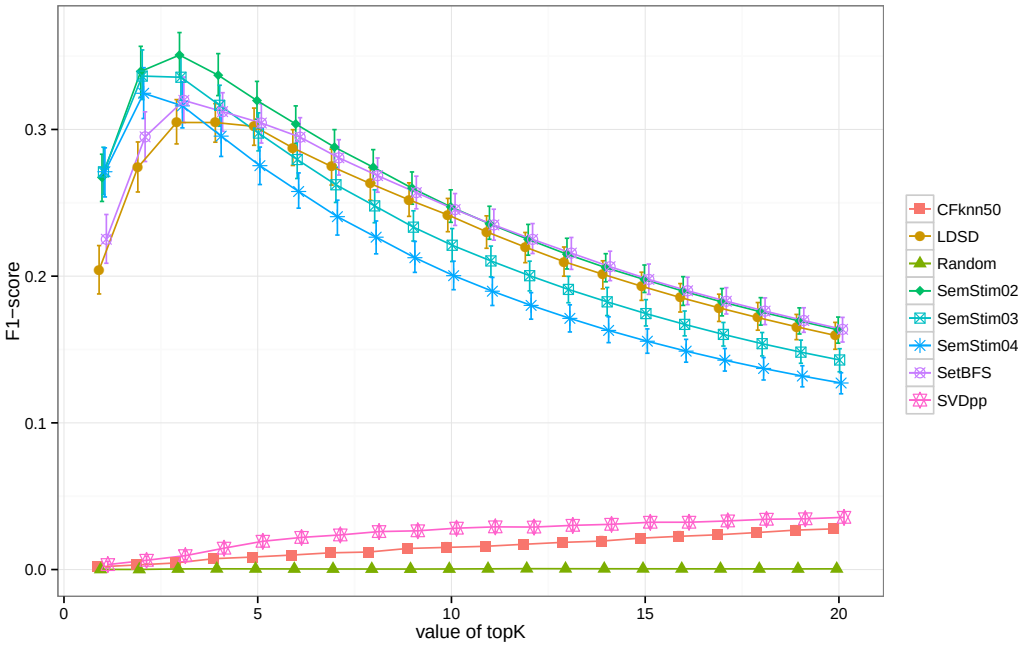

Figure 5.6.: Mean F1-score results for single-domain top-k recommendations, with DVD ratings as source and target domain. Error bars show the standard error.

While SemStim is outperformed by SetBFS and LDSD for this test, this is not related to the sparsity of the data, as SemStim does achieve results for precision, recall and F1-score, which are significantly better then the Random baseline.

For this sparsity test, SemStim with $\tau = 0.2$ has the same performance as SetBFS (the difference is not statistically significant). This points to issues with the quality of the linkage data which connects the items in the SNAP Amazon data to DBPedia.

One reason for quality issues with our linkage data, is that we only used basic heuristics such as fuzzy full-text matching to create this linkage data, as described in Section 5.2.3. In addition, as many entities on DBpedia which correspond to books, music or movies are not correctly typed, many of the generated links point to concepts which have similar names, but which do not represent the correct type of item. For example, a childrens book called "My Very 1st Tea Party" was linked to the general concept of a "Tea party" (a gathering to drink tea).

This can be interpreted as an indicator for the robustness of SemStim to provide recommendations even if the linkage data contains errors. On the other hand, it seems likely that improving the quality of the linkage data, will have a positive impact on the recommendations of SemStim.

Regarding the performance of both CF alghorithms, both SVD++ and CFknn were clearly outperformed by the graph-based algorithm. The performance of the graph-based algorithms is about one order of magnitude better, which suggests that both CF algorithms are unable to use rating data with the given low sparsity. This is understandable, as CF relies on the connections in the user-item matrix, which are too sparse in this data set. On the other hand, the graph algorithms rely on the connections between concepts in the semantic network which indirectly link the source and target domains, so that they are not influenced by the sparsity of rating data.

The sparsity of the DVD rating data is lower than the sparsity of the ratings in the population with 20 ratings for both music and DVDs. Therefore CFknn and SVD++ will have an even worse performance on our cross-domain rating data. For this reason, we do not include CFknn and SVD++ in the cross-domain experiment.

5.4.3. Cross-domain experiment using Amazon SNAP data

After determining the most suitable population from the Amazon SNAP data set, and determining the algorithms which are suitable for the sparsity of the available rating data, we now describe our cross-domain experiment.

We first describe the experimental protocol for generating the cross-domain recommendations, which simply uses one domain as the source domain and the other domain as the target domain. Then we describe the results of the experiment.

5.4.4. Experiment protocol

For our cross-domain experiment, we employ an experiment protocol which is different from both the single-domain accuracy experiment and the single-domain diversity experiment.

In this experiment, we want to evaluate the ability of the participating algorithms to use preferences from one source domain, to provide recommendations for a different target domain. As we use only user profiles with preferences from both "DVDs" and "music", we can use the profile of a user from one domain as the training profile, and the profile of a user from the other domain as the test profile. No further modifications of the training and test profile are necessary. (E.g. there is no splitting of a profile involved, and no random sampling necessary.)

We first describe the results for using DVDs as the source domain and music as the target domain. Then we describe the results when these target and source domains are switched, so that music is the source domain and DVDs are the target domain.

The metrics for the cross-domain experiment are precision, recall and F1-score. For each user, we use all the preferences in the source domain from the training profile to generate the recommendations for the target domain. Then we compare the recommendations with the ground truth from the test profile for each user. Each recommended item which matches an item from the test profile counts as a "true positive". Then we use the number of "true positives" in a recommendation to calculate precision, recall and F1-score.

For SemStim we use the following constants: maximum number of waves: $w_{max} = 5$; maximum number of phases: $\rho_{max} = 4$; default weight of predicates: $\beta = 1.0$; initial node output before applying modifiers: $\alpha = 4.0$; required number of activations in the target domain: $\theta = 30$. For τ we use the values of 0.4, 0.3 and 0.2, and show the results these three levels of τ as part of the results.

We did not include CF in this experiment, as the sparsity of the rating data is too high for the CF algorithm to achieve a better performance then the Random selection baseline. We have described this in detail in Section 5.4.2.

5.4.5. Results for source domain "DVDs" and target domain "music"

The results for the metrics precision, recall and F1-score are in the Figures 5.7, 5.8 and 5.9 respectively.

Figure 5.7 shows the mean precision for levels of k up to 30. SemStim dominates the other algorithms, with its precision peaking at 2.7% for $k = 5$, after which the precision

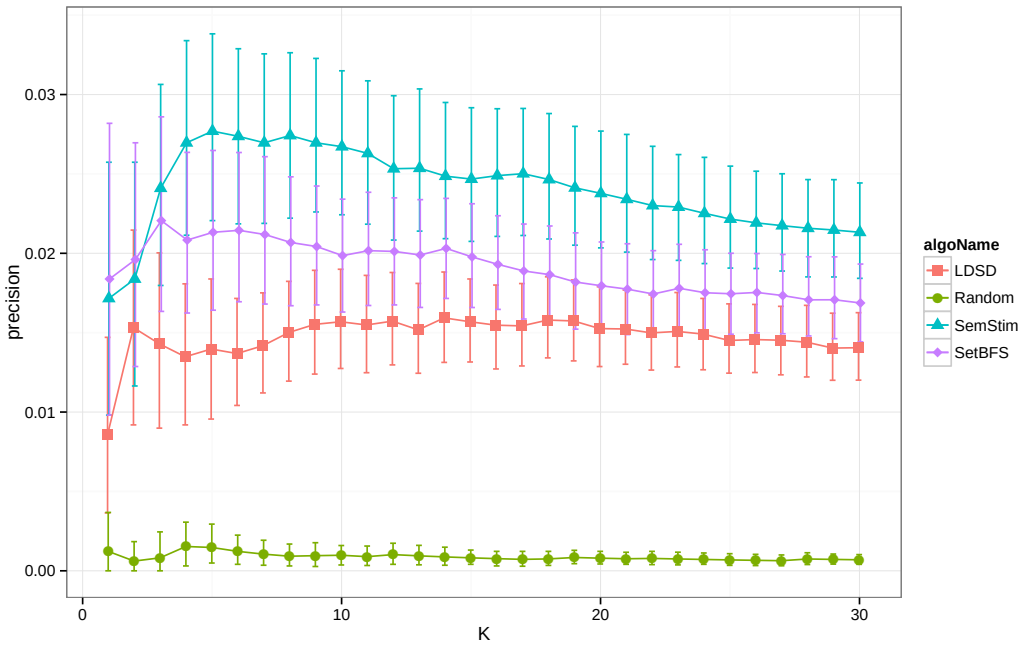

Figure 5.7.: Mean precision results for cross-domain top-k recommendations, with DVDs as the source domain and music as the target domain. Error bars show the standard error.

drops again slightly, down to 2.1% for $k = 30$. The second best performing algorithm is SetBFS with a precision of 1.6% for k=30, followed by LDSD with a precision of 1.4%.

Figure 5.8 shows the mean recall for levels of k up to 30. The recall of all three algorithms is monotonically increasing, with SemStim dominating the other algorithms. For $k = 30$, SemStim has a recall of 1.2%, SetBFS has a recall of 0.8%, and LDSD has a recall of 0.7%.

Figure 5.9 shows the mean F1-score for levels of k up to 30. All three algorithms show a stable trend, with SemStim having the best performance. For $k = 30$, SemStim has an F1-score of 1.3%, SetBFS has an F1-score of 1.0%, and LDSD has an F1-score of 0.8%.

While SemStim performs better than LDSD and SetBFS for all three metrics, the overall results for all three metrics are quite low, which can be explained by the high sparsity (99.9883%) of the ratings data for the population of users with 20 ratings for both “DVDs” and “music”.

The distance between the results of the three algorithms is relatively small for all three metrics. The error bars overlap for all levels of k. In addition, as discussed in the sparsity analysis of the Amazon SNAP data set, the population size is quite small. So we performed a pairwise Wilcox test, to determine if the results are statistically significant. We determined that for high levels of $k > 20$, the difference between SemStim and both baselines is statistically significant with $p < 0.01$.

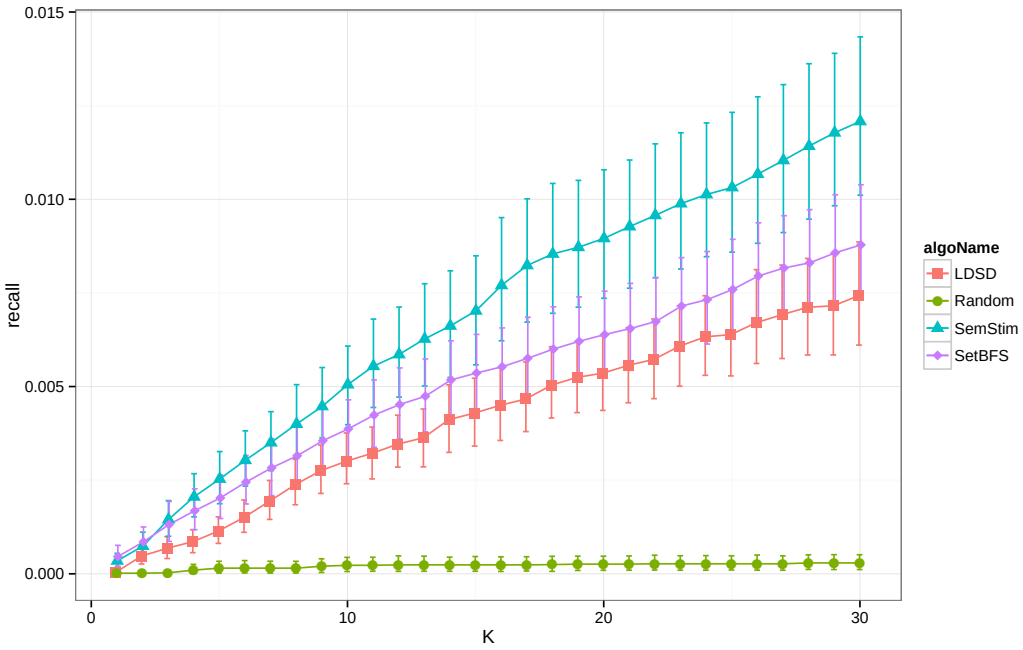

Figure 5.8.: Mean recall results for cross-domain top-k recommendations, with DVDs as the source domain and music as the target domain. Error bars show the standard error.

In summary, when using the source domain DVDs and the target domain music for the cross-domain recommendation task, the performance of SemStim dominates that of the other two graph algorithms.

5.4.6. Results for source domain "music" and target domain "DVDs"

As we have shown, Semstim performs better than the other two graph algorithms, when using DVDs as the source domain and music as the target domain. However, when we switch the source and target domains, SemStim only performs as well as SetBFS.

Figure 5.10 shows the mean precision for levels of k up to 30. The best performing algorithm is SetBFS with a precision of 3.7% for $k = 30$. However SemStim with $\tau = 0.2$ performs as well as SetBFS, as the difference between the precision of the two algorithms is not statistically significant for $k > 20$. SemStim with $\tau = 0.3$ and $\tau = 0.4$ perform worse, as does LDSD. The worst performing algorithm is the random baseline.

Figure 5.11 shows the mean recall for levels of k up to 30. The recall of all three algorithms is monotonically increasing. SemStim with $\tau = 0.2$ is dominating the other algorithms. This is followed by SemStim with $\tau = 0.3$, SetBFS, SemStim with $\tau = 0.4$

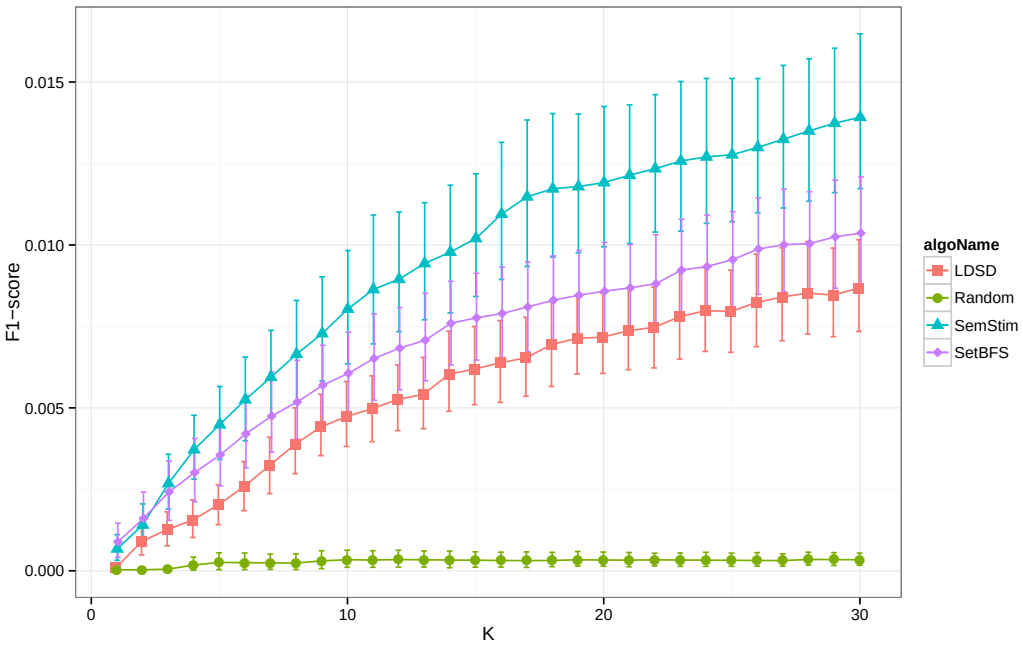

Figure 5.9.: Mean F1-score results for cross-domain top-k recommendations, with DVDs as the source domain and music as the target domain. Error bars show the standard error.

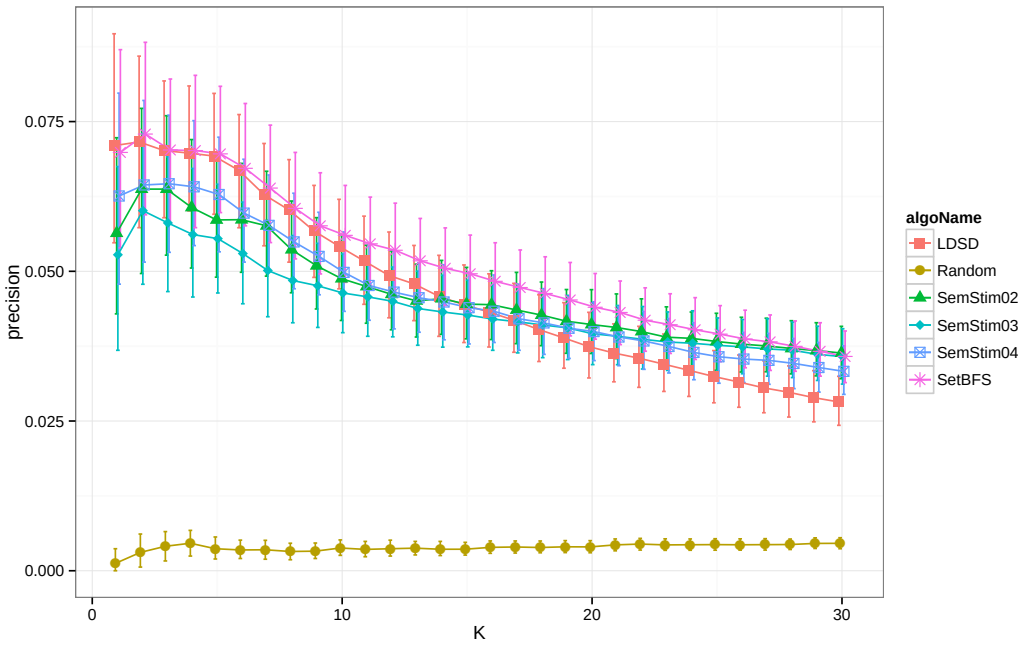

Figure 5.10.: Mean precision results for cross-domain top-k recommendations, with music as the source domain and DVDs as the target domain. Error bars show the standard error.

and LDSD. However, for $k > 20$ the difference between SetBFS and all variations of SemStim is not statistically significant for recall.

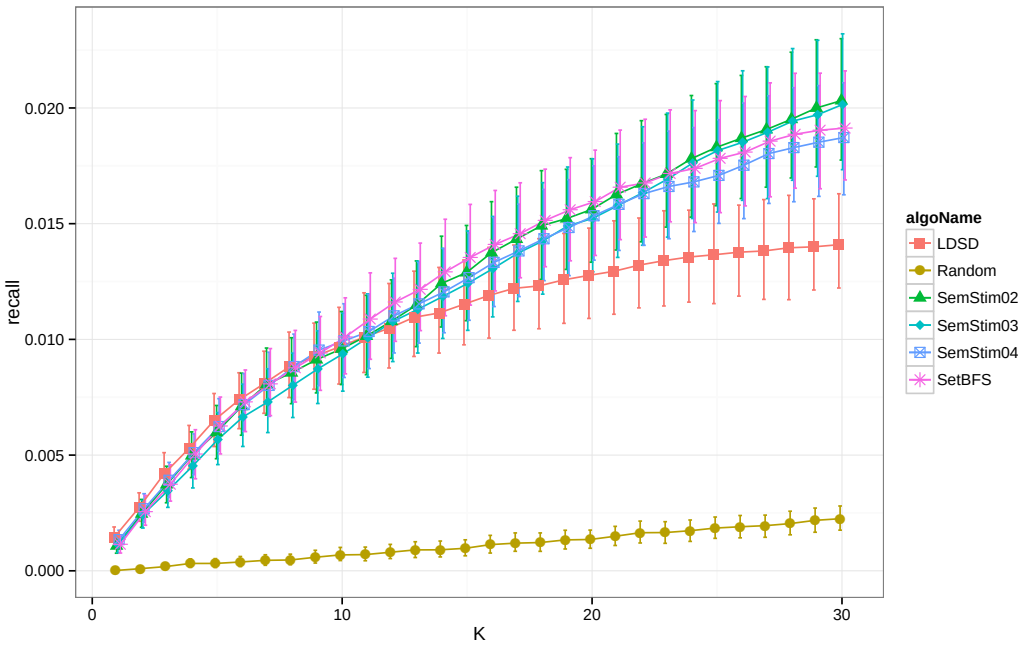

Figure 5.11.: Mean recall results for cross-domain top-k recommendations, with music as the source domain and DVDs as the target domain. Error bars show the standard error.

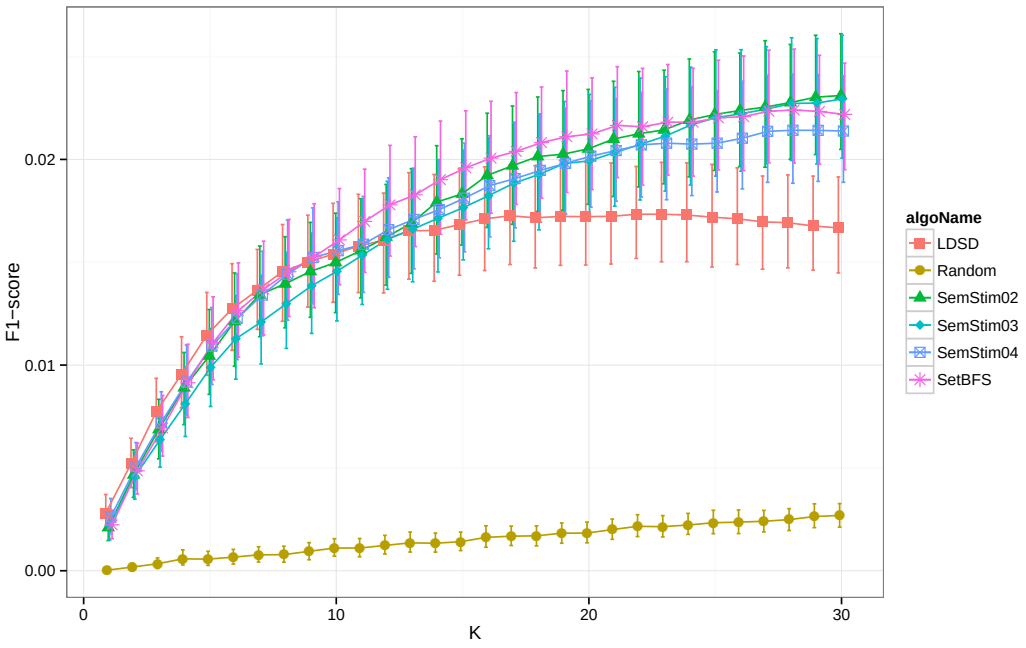

Figure 5.12.: Mean F1-score results for cross-domain top-k recommendations, with music as the source domain and DVDs as the target domain. Error bars show the standard error.

Figure 5.12 shows the mean F1-score for levels of k up to 30. This shows the same trends as for recall. SemStim with $\tau = 0.2$ is dominating the other algorithms, followed by

SemStim with $\tau = 0.3$, SetBFS, SemStim with $\tau = 0.4$ and LDSD. Again, for $k > 20$ the difference between SetBFS and all variations of SemStim is not statistically significant.

In summary, after reversing target and source domains from the previous experiment, the performance of SemStim does not outperform SetBFS. Instead SemStim with $\tau = 0.2$ only performs as well as SetBFS, with the difference between the two algorithms being not statistically significant.

5.4.7. Discussion

The results of our cross-domain experiment show that SemStim can be used for the intended purpose of generating cross-domain recommendations.

When using DVDs as the source domain and music as the target domain, SemStim outperforms the other algorithms with statistically significant results. However, when reversing the source and target domain, so that music is used as the source domain and DVDs as the target domain, SemStim performs only as good as SetBFS, with a not statistically significant difference between the performance of the two algorithms.

The results of the cross-domain accuracy experiment have shown, that pairs of domains exist for which the performance of SemStim is not symmetrical. In other words, SemStim performs better when using one of two domains as the target domain, and it might perform slightly worse, when the target and source domains become switched. In our experiment, SemStim performed better then the comparison algorithms when using DVDs as the source domain, and music as the target domain. When these target and source domains were switched, SemStim performed only as good as SetBFS, with the difference between the two algorithms being statistically insignificant.

This indicates that the performance of SemStim is dependent on the selection of the specific target and source domain. This raises the question, about which characteristics of the selected domains influence the performance of SemStim. Possible characteristics of a domain might be the connectedness of a domain, or the size of a domain. As future work, we plan to investigate the connection between the performance of SemStim and the choice of target and source domain.

We now discuss the limits of our cross-domain experiment, which lie in the small data set size, the high sparsity of the data, and the overall low performance of all algorithms for precision, recall and F1-score. We also discuss the limitations of the data set which we used in the experiment.

In our cross-domain experiment we used the ratings of the population of users with 20 ratings for both music and DVDs. This reduces the number of eligible users from 1,389,641 down to 845 users. The remaining users have rated 51,657 items from these two domains, out of a total number of 122,972 items. We had to choose this population of

users, as we required each user to have preferences in two domains in order to be able to use the data for a cross-domain experiment. In addition, we required 20 rated items for each of the two domains, in the same way that the MovieLens 100k rating data requires 20 ratings of each user. Therefore the data which we use in our cross-domain experiment is from the largest suitable population of users from the Amazon SNAP data set.

The second limitation of our experiment is the high sparsity of the rating data used. As we showed in our sparsity analysis of the Amazon SNAP data, the rating data which we use has a sparsity of 99.9883%, meaning that the user-item rating matrix only contains a fraction of all possible ratings. However, this is the lowest available sparsity for any population of users with 20 items rated in two domains.

To test if this high sparsity has an impact on the algorithms under evaluation, we performed a test experiment. For this experiment we used the DVD ratings from the selected user population, and the same experiment protocol as in the single-domain accuracy experiment. This makes the results of the sparsity test comparable to the single-domain accuracy results. The results of the sparsity test showed that all graph-based algorithms were able to provide recommendations. However, the CFknn algorithm provided results which were as bad as the Random selection baseline. In summary, with the sparsity test we showed that all three graph-based algorithms are suitable to the low sparsity of the rating data, because they are, in effect similarity based algorithms which are unaffected by user item sparsity.

Due to the small size of the data set and the high sparsity of the data set, the overall results for the metrics of precision, recall and F1-score are quite low, when compared to results e.g. in the field of Information Retrieval. However, the goal was to show that SemStim is able to perform the cross-domain recommendation task with different target and source domains. While the results are low, they do support this conclusion.

Our evaluation also points to the lack of publicly available data sets with cross-domain user profiles. In our experiment we used the "Amazon product co-purchasing network metadata" set, which is provided by the Stanford Network Analysis Project (SNAP) as described in Section 5.2.3. However, this data set was collected for the purpose of research on the topic of viral marketing in social networks. As such, the main limitation of the data set which we used, is that only 295,408 (21%) of the 1,389,641 user profiles contain one or more ratings in at least two domains. This leaves 1,094,233 (79%) user profiles which do not contain any cross-domain preferences, and as such do not provide any ground truth for this cross-domain experiment.

In summary, although our cross-domain experiment has clear limitations, it shows that SemStim is the most suitable of the evaluated algorithms for performing cross-domain recommendations.

5.5. Single-Domain Diversity Experiment

After having evaluated the accuracy of SemStim for both the single-domain and the cross-domain recommendation task, we additionally want to evaluate diversity as a different aspect of the SemStim recommendation results.

Increasing the diversity of recommendations can lead to higher user satisfaction and engagement according to [Winoto & Tang, 2008]. In some cases suggesting a set of very similar items may not be as useful for the user, because it may take longer to explore the range of available items according to [Shani & Gunawardana, 2011]. Recommending more diverse items in turn can allow the user to get a quicker overview of the whole range of available items.

In our diversity experiment, we want to evaluate and compare the diversity distributions for the recommendations generated by different algorithms. In addition, we want to evaluate if the activation threshold of SemStim can be used to tune the recommendations to be more or less diverse.

For our experiment we use our own cluster-based diversity metric, which estimates the number of clusters in a set of DBpedia URIs in order to quantify the diversity of the set.

Our cluster-based diversity metric allows us to make observations about the expected diversity of recommendations generated by each algorithm. Each diversity distribution shows if an algorithm provides (a) very diverse sets of recommendations, or (b) very similar sets of recommendations, or (c) if an algorithm is somewhere in the middle of this spectrum. Consider the following examples:

- A diversity distribution with a small range of diversities with a median close to 1.0, would indicate that an algorithm consistently makes sets of recommendations which are very diverse.
- A diversity distribution with a small range of diversities with a median close to 0.0, would indicate that an algorithm consistently makes sets of recommendations which a very similar to each other.
- A diversity distribution with a large range of diversities with a median close to 1.0, would indicate that an algorithm can make both very similar and very diverse sets of recommendations.

In this section, we will first describe the experiment protocol of our single-domain diversity experiment. Then we describe our diversity metric, as well as several other unsuitable diversity metrics which we considered for our experiment. Then we present our results and discuss the results.

5.5.1. Experiment protocol

As we want to compare the diversity distributions of SemStim with all other algorithms including CFknn and SVD++, we use MovieLens 100k rating data as both the source *and* target domain.

We use the following experiment protocol for our single-domain diversity experiment: First we split the user profile of the active user into training profile and test profile. For this, we sample 5% of the most highly rated items of the user and place these items in the test profile of the user. The remaining 95% are placed into the training profile of the active user.

Then we use the training profile of each user to generate recommendations with each algorithm in the same way as described in Section 5.3.1 of the single-domain accuracy experiment. However, instead of setting the target domain to the test profile of the active user as in the accuracy experiment, we set the test profile to the set of all movies in the MovieLens 100k rating data, which are unrated by the active user. This allows each algorithm to generate recommendations without any additional constraints.

After generating recommendations with each algorithm, we apply our diversity metric to each set of recommendations generated by one algorithm.

The outcome of this experiment is a distribution of diversity values for each algorithm. As explained, this allows us to compare the diversity distributions between algorithms, and to make observations about the expected diversity of recommendations generated by an algorithm.

We chose this experiment protocol, as it does not put any constraints on the recommended items, beyond requiring them to be from the MovieLens 100k rating data, which only contains a subset of 1,682 movies from all 72,000 movies in DBpedia. This allows us to make better observations about the diversity of the different algorithms, when the target domain is restricted to the items in the inventory of a recommender system. The inventory of a recommender system is usually limited to the items which are physically in stock (e.g. Amazon) or which have been licensed by the operator of the recommender system (e.g. Netflix).

For SemStim we use the following constants: maximum number of waves: $w_{max} = 5$; maximum number of phases: $\rho_{max} = 4$; default weight of predicates: $\beta = 1.0$; initial node output before applying modifiers: $\alpha = 4.0$; required number of activations in the target domain: $\theta = 30$. In addition, we set the activation threshold τ to all values from 0.1 to 1.5 in increments of 0.1, which allows us to see if the diversity is correlated to the activation threshold. We set the maximum number of waves and phases to high values of 5 and 4 respectively, in order to allow SemStim to search a bigger subset of the semantic network. All other parameters have the same value as in the single-domain accuracy experiment.

5.5.2. Unsuitable similarity measures

We first considered several other metrics for quantifying the diversity of movies from DBpedia, which proved to be unsuitable. [Panchenko, 2011] lists 6 graph-based similarity metrics, which are the Inverted Edge Count, the Leacock and Chodorow measure, the Resnik measure, the Jiang and Conrath measure, and the Wu and Palmer measure. These other measures are arguably simpler to implement then our cluster-based approach.

However, each of these graph-based similarity measures was not able to distinguish between very diverse or very similar sets of movies from DBpedia. We used each of the 6 graph-based similarity metrics to determine the average intra-list similarity of (a) the recommendations for all users by all 6 algorithms, (b) artificially constructed sets of movies.

The artificially constructed sets of movies were picked by hand. The movies in these sets were selected in order to either make the final set of movies be very diverse or very similar. We then used the artificial sets of movies to test if the 6 graph-based similarity metrics from [Panchenko, 2011] are sensitive enough to distinguish between very diverse or very similar movies. The sets of very similar movies contained e.g. all Star Wars or all Harry Potter movies. The sets of very diverse movies contained e.g. 10 different Art-House movies, each from a different European or Asian country.

However, for the artificial sets of movies, all 6 graph-based similarities were not able to distinguish between the diverse and similar sets of movies. For the real recommendations, for each algorithm, the average intra-list similarity was the same for all recommendation algorithms.

The failure of the graph-based similarities to quantify the diversity of sets of movie URIs, can be explained by problematic features of the three kinds of properties which distinguish movies on DBpedia. These properties come from either the DBpedia Ontology, the YAGO ontology or the SKOS categories.

The DBpedia Ontology is not consistent enough in applying properties to entities of the same kind. E.g. not all movies are typed as `dbpedia-owl:Film`. This makes the DBpedia ontology to unreliable to be used in a graph-based similarity measure.

The YAGO Ontology is very shallow when it comes to movies. There are special types of movie, such as `yago:CyberpunkFilms`, however these are all direct descendants of only one super-node `yago:Movie106613686`. This makes the yago ontology unsuitable for any distance based similarity.

The SKOS categories have an unsuitable topology for the task of graph-based similarity, as the graph of SKOS categories contains loops, while most graph-based similarities require a strict hierarchy.

Due to these problematic features of the different kinds of properties of movies on DBpedia, we could not use a graph-based diversity in our experiment. Instead we developed our own feature-based diversity.

5.5.3. Cluster-based diversity metric

Our diversity metric is based on estimating the number n of clusters in a set R of recommendations, where a set of recommendations with more clusters is more diverse than another set with less clusters.

Our approach to quantifying the diversity of movie URIs from DBpedia is not graph-based. Instead it first uses the properties of a movie as features. Then the cosine distance between the feature vectors is used to construct a distance matrix. This matrix then is used to estimate the number of clusters in the set of URIs.

We quantify the diversity of a set of movie URIs from DBpedia in five steps:

1. We calculate the matrix of pairwise distances between all items in R. We use the cosine distance between movie feature vectors. The RDF types and SKOS categories of each movie are used as features. These are all the objects of triples connected to the movie via the properties `rdf:type` and `dcterms:subject`. We remove outlying features which are shared between more then 15% of all movies, and features which are shared by less then 2% of movies. The features are weighted according to the term frequency-inverse document frequency (TFIDF) [Jones, 1972], with movies as documents and features as terms.

2. We perform a hierarchical clustering (cf. [Xu et al., 2005]) on the distance matrix using the median distance as the agglomeration criterion.

3. Then we evaluate all cluster numbers (2 to $|R|-1$), which can result from the hierarchical clustering, using three different cluster evaluation metrics [Halkidi et al., 2001]. These are (a) the silhouette coefficient, (b) the Dunn index and (c) the Goodman and Kruskal index with parameter $g = 3$. These cluster evaluation metrics indicate the optimal number of clusters for a particular set of items. Each cluster evaluation metric will have its maximum value for one cut of the hierarchical clustering.

4. The final cluster estimate is calculated as the average of agreeing cluster evaluation metrics. We define agreeing cluster evaluation metrics as returning a cluster estimate within 25% of the whole range of possible cluster estimates.

5. As the last step, we normalise the cluster estimate into a diversity score between 0 and 1, where 1 is the maximum diversity.

The resulting diversity metric is successfully able to distinguish between diversity and similar artificial sets of movies, and it also is able to distinguish the results generated by the different algorithms.

[Vargas & Castells, 2011] present a systematic framework for quantifying novelty and diversity of recommendations. Vargas et al. distinguish between two main approaches for quantifying novelty and diversity, which are either based on item popularity or the distance between items. When using the popularity of an item, then less popular items have a higher novelty. As an alternative to considering the popularity of an item, Vargas et al. identify the class of distance-based metrics which use a similarity metric to quantify the distance between items. Then maximising the average intra-list distance between a list of items leads to a maximum of diversity between the items. Our approach can be classified as a distance-based diversity metric in the framework of Vargas et al.

5.5.4. Results

We now describe the results for our single-domain diversity experiment.

As we will show, when only using positive ratings of a user for SemStim, there is no correlation between the diversity of the recommendations generated by SemStim. However, when all the ratings (positive, neutral and negative) of a user are utilised by SemStim, there is a strong connection between the diversity and the activation threshold.

We first describe the results for SemStim when only positive ratings are used. Then we describe the results for using SemStim with all ratings of a user.

Figure 5.13 shows the distributions of diversity values for all algorithms (CFknn, Random, LDSD, SetBFS and SemStim), and for all levels of the activation threshold τ for SemStim. For these results, SemStim only used positive ratings to generate the recommendations. The dotted line separates SemStim from the comparison algorithms. The highlighted box-plot shows the diversity distribution for setting the activation threshold $\tau = 0.4$, as in the single-domain accuracy experiment.

CFknn and the random baseline produce recommendation sets with the highest median diversity, followed by LDSD. SetBFS generates has a slightly smaller median, and SVD++ has the lowest median diversity of the comparison algorithms. For SemStim, the median diversity values for all levels of τ are very similar.

The range of diversity values is smallest for CFknn50 and Random selection. For the graph-based algorithms, LDSD and SetBFS, the range of diversity values increases slightly. SVD++ has the broadest range of diversity values of the comparison algorithms.

The diversity distributions for these algorithms show that we can expect CFknn50, LDSD and Random to consistently provide only diverse recommendations. SetBFS

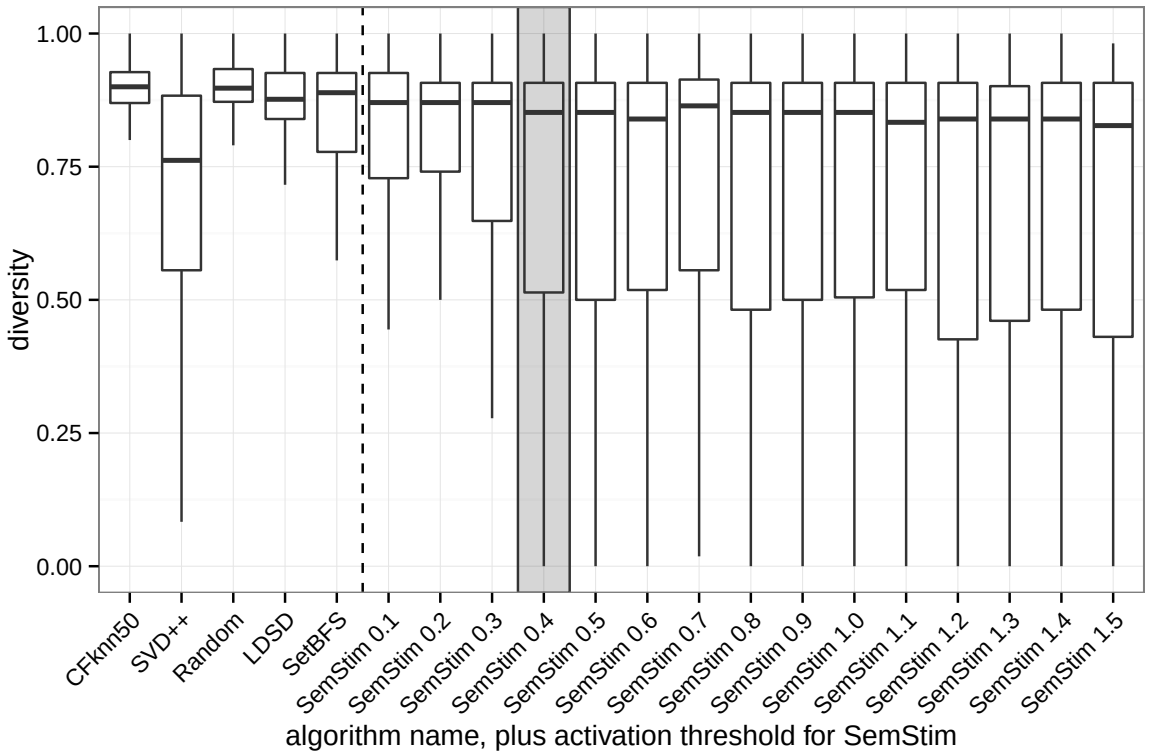

Figure 5.13.: Diversity distributions for recommendations. The SemStim results are based on the *positive* ratings of a user. The highlighted area shows the diversity distribution for SemStim with $\tau = 0.4$. This value for τ is also used in the single-domain accuracy experiment.

provides recommendations which can be expected to be on the upper end of the diversity scale. SVD++ can provide the broadest distribution of diversities of the comparison algorithms.

However, for SemStim we can observe that for $\tau \geq 0.4$ there is no difference between the range of diversity values in the diversity distributions. This indicates that there is no correlation between the level of the activation threshold τ and the range of diversities for all recommendations generated by SemStim.

We used Kendall's rank correlation to analyse the correlation between the τ and the standard deviation of the diversities of the recommendations. We use the standard deviation as a proxy for the "broadness" of a diversity range. This correlation analysis showed that the standard deviation of diversities is not correlated to the activation threshold τ when using only positive user preferences for SemStim.

However, when using all preferences of a user, the activation threshold τ can be used to tune the range of diversities for SemStim. Figure 5.14 shows the diversity results for SemStim when all ratings of a user are used.

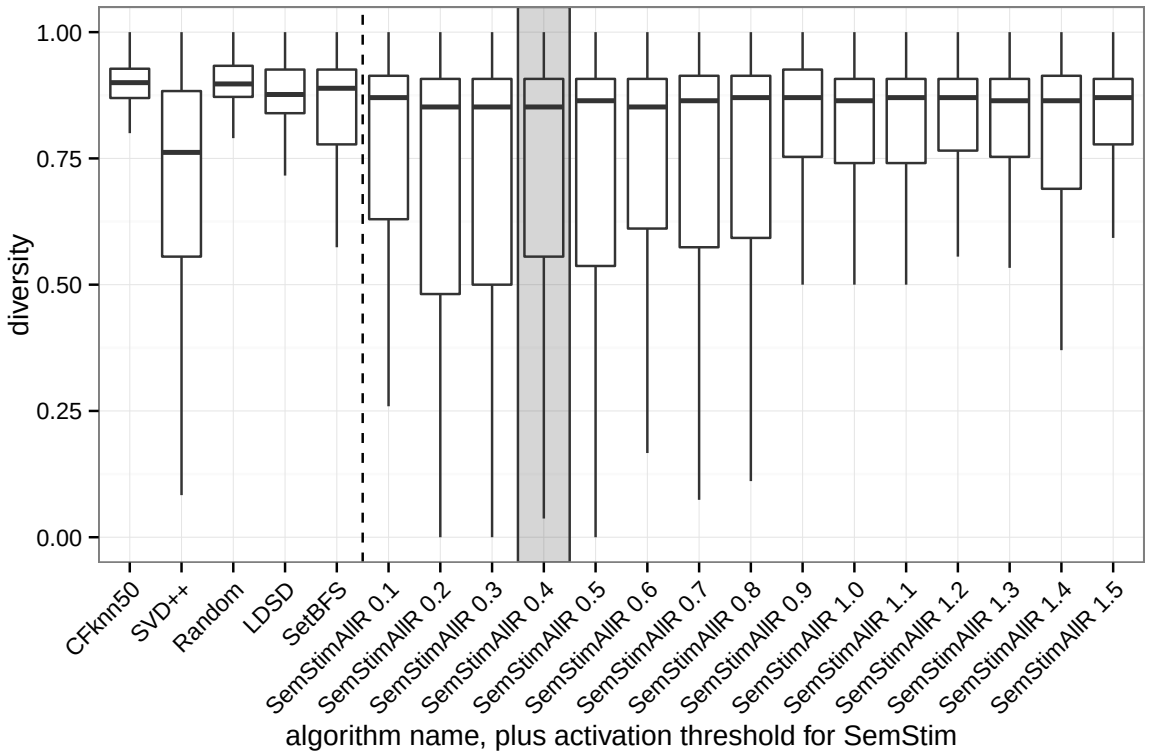

Figure 5.14.: Diversity distributions for recommendations. The SemStim results are based on *all* ratings of a user (positive, neutral and negative). The highlighted area shows the diversity results for SemStim with $\tau = 0.4$. This value for τ is also used in the single-domain accuracy experiment.

We can easily see, that different levels of τ result in SemStim having very different diversity distributions. The range of diversity values is "broadest" for $0.2 \leq \tau \leq 0.5$. Then the diversity distributions get narrower for $0.6 \leq \tau \leq 0.8$. More narrow diversity distributions are achieved for $0.9 \leq \tau \leq 1.4$. Finally, the most narrow diversity distribution is achieved for $\tau = 1.5$.

τ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
std. deviation	0.275	0.327	0.334	0.317	0.325	0.291	0.310	0.283	0.264

τ	1.0	1.1	1.2	1.3	1.4	1.5
std. deviation	0.260	0.263	0.258	0.257	0.267	0.237

Table 5.1.: Standard deviation of SemStim diversity values for all levels of the activation threshold τ from 0.1 to 1.5

We can use the standard deviation of the diversity distributions of SemStim to quantify the range of diversity values for the different levels of the activation threshold. Table 5.1 shows the standard deviation of the diversity values for all levels of the activation

threshold. The standard deviation is highest for $\tau = 0.3$ at 0.334 and for $\tau = 0.2$ at 0.327. It is lowest for $\tau = 1.5$ at 0.237.

For SemStim we can see the trend of reducing diversity range with increasing activation threshold. We verified this trend by performing Kendall's rank correlation analysis between the activation threshold and the standard deviation. We use the standard deviation as a proxy for the "broadness" of a diversity distribution. The result of Kendall's rank correlation analysis supports the hypothesis of a negative correlation with $p = 0.0001$ and Kendall-$\tau = -0.695$.

The result of the correlation analysis suggest that there is a connection between the activation threshold τ and the range of diversity distributions: Setting a higher value for τ leads to recommendations with a high diversity. Conversely, using a lower setting for τ results in recommendations which are somewhere in the spectrum between diverse and similar.

While this connection between the activation threshold τ and the "broadness" of the diversity distribution is not linear, it can be used to tune the recommendations of SemStim to be either more diverse or more similar.

5.5.5. Discussion

In our diversity experiment, we have compared the diversity distributions of both graph-based and non-graph-based algorithms. In addition, we have evaluated the connection between different levels of the activation threshold τ on the diversity of the results, which are generated by SemStim.

For our experiment, we propose a novel diversity metric that is based on estimating the number of clusters in a set of DBpedia URIs. Our cluster-based diversity metric allows us to make observations about the expected diversity of recommendations generated by each algorithm.

For our experiment we use a single-domain experiment protocol. This allows us to compare the diversity distributions of SemStim with the diversity distributions of the other graph-based algorithms and the collaborative filtering algorithms. We used the MovieLens 100k rating data as the source and target domain.

The results of this experiment show that the recommendations of SemStim can be tuned between being more diverse or more similar, by changing the activation threshold of nodes, which is one of the parameters of the algorithm. However, tuning the recommendations this way, requires using all of the preferences of a user, including his negative and neutral user preferences. When only using the positive preferences of a user, there is no correlation between the range of diversity and the activation threshold.

Regarding the influence of the activation threshold τ on diversity, we can see the trend of reducing diversity range with increasing activation threshold. We verified this trend by performing Kendall's rank correlation analysis between the activation threshold and the standard deviation of SemStim diversity distributions, which supports the hypothesis of a negative correlation with $p = 0.0001$ and Kendall-$\tau = -0.695$.

The result of the correlation analysis suggest that there is a connection between the activation threshold τ and the range of diversity distributions: Setting a higher value for τ leads to recommendations with a high diversity. Conversely, using a lower setting for τ results in recommendations which are somewhere in the spectrum between diverse and similar.

While this connection between the activation threshold τ and the "broadness" of the diversity distribution is not linear, it can be used to tune the recommendations of SemStim to be either more diverse or more similar.

The main limitation of our diversity experiment, is that the experiment protocol limits our experiment to make observations about diversity, without being able to make any observations about accuracy. However, we investigate the relationship between diversity and accuracy of SemStim in Section 5.6, as part of the description of our results for the diversity task at the Linked Open Data-enabled recommender systems challenge.

In summary, the results of our single-domain diversity experiment show that SemStim can provide recommendations with very diverse or very similar recommendations, if all ratings in a user profile are used. The diversity of recommendations provided by SemStim can be tuned by the activation threshold.

5.6. Linked Open Data-enabled Recommender Systems Challenge

In this section we describe the results of our participation with SemStim at the "Linked Open Data-enabled Recommender Systems" (LODRecSys) challenge at the Extended Semantic Web Conference (ESWC) 2014.

The goal of participating in this challenge with SemStim was two-fold:

1. we wanted to show that SemStim has a competitive performance for the single-domain diversity recommendation task of the challenge, despite the disadvantage of being designed for the requirements of cross-domain recommendation. The challenge allowed us to directly compare our results to those of all other participants.

2. We aimed to investigate the relationship between accuracy and diversity of the recommendations generated by SemStim. The diversity recommendation task was

especially designed to reward approaches which are able to balance accuracy and diversity, as each approach is ranked using a combination of F1-score and intra-list diversity.

We first describe the challenge and its evaluation tasks in detail. Then we describe how we used SemStim in the challenge, and we describe the results of our participation.

5.6.1. Details of the challenge and the evaluation tasks

The LODRecSys challenge[14] is based on the same ideas as other challenges in the recommender systems research community, like the famous Netflix prize [Bell & Koren, 2007]. The LODRecSys challenge has three main features, which are (1) the competition between teams, (2) real-time result submission and (3) secret ground-truth for the test data.

In order to participate in the challenge, teams need to sign-up using a special web site. Then each team could download the training data of the challenge, as well as the test data. However, the features of the test data, which are required to calculate the metrics, have been removed and are only known to the organisers of the challenge. Each team then implements one or more of the evaluation tasks.

The organisers define a result submission format for each evaluation task. The participating teams submit the results of their recommender systems to the evaluation system of the challenge. The result submission enforced that the results file to contain at least 20 recommendations for each user. It was not possible to provide no recommendation or less than 20 recommendations for a user.

The evaluation system uses the submitted results file and the secret ground truth to calculate the metrics directly after submitting a result. The resulting metrics are displayed for the submitting team. The evaluation system only reports aggregate metrics for each result submission. In other words, each time a team submits a result, it will receive exactly one result for each metric. It is not possible to get the results of the metrics for e.g. a single user or for a group of users.

In addition, the evaluation system of the challenge provides the overall ranking of participating teams for each evaluation task. The challenge itself runs over multiple weeks, and the team which has the best results for each evaluation task receives a prize at the end.

As the ground-truth is unknown to the participating teams, the only real performance baseline for participants is the performance of the other teams. This makes an evaluation challenge a good environment for trying out new forms of recommendation algorithms. In the case of the LODRecSys challenge, the organisers focused on content-based recommender systems using Linked Data. This is reflected in the choice of the data set, and

[14] http://challenges.2014.eswc-conferences.org/index.php/RecSys

in the choice of evaluation tasks. The data set is the DBbook data set, which we have described in 5.2.4. The details of the three evaluation tasks are as follows:

Rating prediction task: This task provides rating data on a scale of 0 to 5 stars as the training data. The test data contains user IDs and item IDs. The ratings are withheld from the test data, and the recommender system has to predict the ratings. The metric used for this evaluation task is the round mean square error (RMSE) of all the predicated ratings and the true ratings.

Top-N recommendation task: This task requires recommending the top-5 items from the test user profile of a user, where the ratings are withheld. The training data contains binary ratings, where 1 indicates a preference, and 0 indicates an irrelevant item. The test data contains user IDs and item IDs but no ratings. The metric used for this evaluation task is the F1-measure@5.

Diversity recommendation task: This task requires recommending the top-20 of all unrated items for a user. The training data is the same as for the top-N recommendation task. No test data is provided, as the recommendations need to be made using all unrated items. This task uses a combination of two metrics, the F1-measure@20 and the Intra-List Diversity @20 (ILD@20). When submitting a result for this task, both the F1@20 and the ILD@20 are determined for the submitting team. Then the rank of the team is determined for each metric, and the overall rank of the team for the diversity task is the average of F1@20 rank and the ILD@20 rank. In order to achieve a high overall rank for this task, the ranks for both F1@20 and ILD@20 need to be high. Achieving a high rank only for one of the metrics is not sufficient.

All metrics used for the challenge are described on the web site[15] of the challenge.

5.6.2. Selection of evaluation tasks

The challenge provides data for 3 evaluation tasks, which are (1) the rating prediction task, (2) the top-N recommendation task and (3) the diversity recommendation task. As we now explain, SemStim is only applicable to the diversity recommendation task.

SemStim is not applicable to the *rating prediction task*, as our algorithm does not make use of rating data. In addition, SemStim could not perform the *top-N recommendation task*, given the properties of the evaluation data set. We analysed the test data set, and we found that the median number of items in the test profiles is 12. In order to perform the given top-N recommendation task, the items in the test profile need to be ranked and the top-5 items need to be recommended. SemStim would need to start with the train profile of a user and then find at least 5 books from the 12 books in the test profile in a

[15] http://sisinflab.poliba.it/semanticweb/lod/recsys/2014challenge/eswc2014-lodrecsys-metrics.pdf

graph with 11 million vertices. Initial testing showed that this exceeds reasonable limits for runtime and memory of our current implementation so we decided that participation in the top-N recommendation task was not feasible with the time limits available to us.

However, we were able to apply SemStim to the given *diversity recommendation task*. In order to perform the diversity recommendation task, a list of top-20 recommendations needs to be made, using the unrated books of a user. As the DBbook data set contains 6733 books, this amounts to using SemStim to find 20 of 6733 books on the DBpedia graph for 6181 users. Testing showed that using the available data, SemStim can perform the diversity recommendation task for all 6181 users.

5.6.3. Applying SemStim to the diversity recommendation task

We use SemStim to generate 20 recommendations for each user profile in the training data of the diversity recommendation task in 6 steps, as follows:

1. We determine the set P of start nodes for the active user. The training data contains binary ratings, where 1 indicates relevance for the user and 0 indicates irrelevance. If the user has any positive preferences, then their corresponding DBpedia URIs are added to P, while ignoring all negative preferences. However, if the user has no positive preferences at all, then the DBpedia URIs of all his negative preferences are added to P. 1463 users have only negative preferences in their user profiles.
2. We set the target domain T to be the set of DBpedia URIs for all unrated books of the active user.
3. We set the constants of SemStim. As we describe later in this section, we experimentally determined that we achieve the best results for the following configuration: activation threshold: $\tau = 0.7$; maximum number of waves: $w_{max} = 1$; maximum number of phases: $\rho_{max} = 5$; required number of activated nodes from T: $\theta = 20$; default weight of predicates: $\beta = 1.0$; initial node output before applying modifiers: $\alpha = 4.0$.
4. Then we run the algorithm. When the algorithm terminates, we rank the activated nodes by their activation value in descending order, and return the corresponding DBbook IDs of the first 20 nodes.
5. If the number of activated nodes is less than 20, we add random, unrated items to reach the target size of 20, as the evaluation system requires all users to have 20 recommendations.

5.6.4. SemStim Results for the diversity recommendation task

The result which we submitted at the end of the challenge uses 5 phases, 1 wave and an activation threshold of 0.7. We arrived at this configuration of the algorithm in multiple steps.

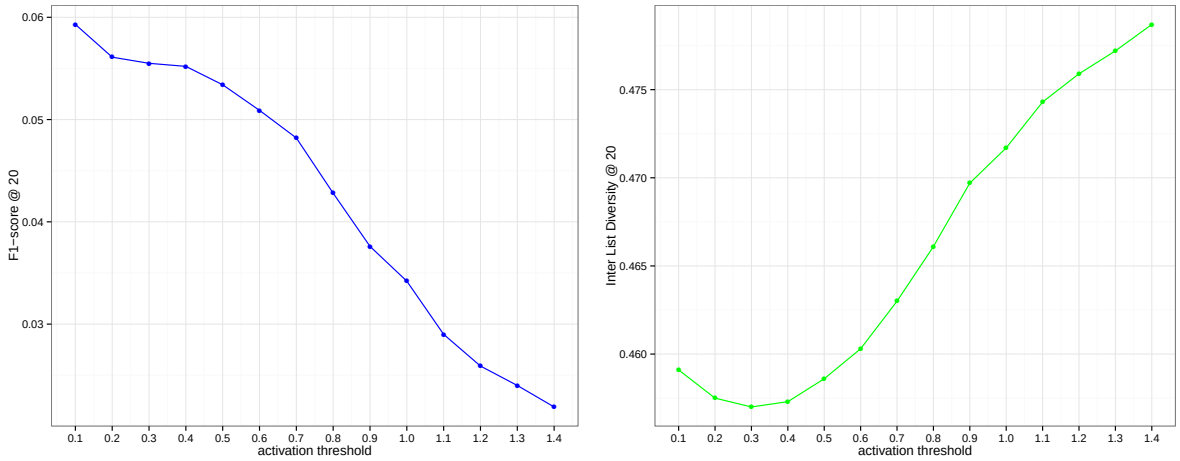

Figure 5.15.: Scores for F1@20 and ILD@20 for activation threshold from 0.1 to 1.4, with 4 phases and 3 waves.e

First, we started by fixing the configuration at 4 phases and 3 waves. However, we submitted results for all activation thresholds from 0.1 to 1.4 in increments of 0.1. Figure 5.15 shows the resulting scores for F1@20 and ILD@20. At the time of submitting the results, we achieved the best ranking among our submitted results, using an activation threshold of 0.6, with 0.0509 F1@20 and 0.4603 ILD@20.

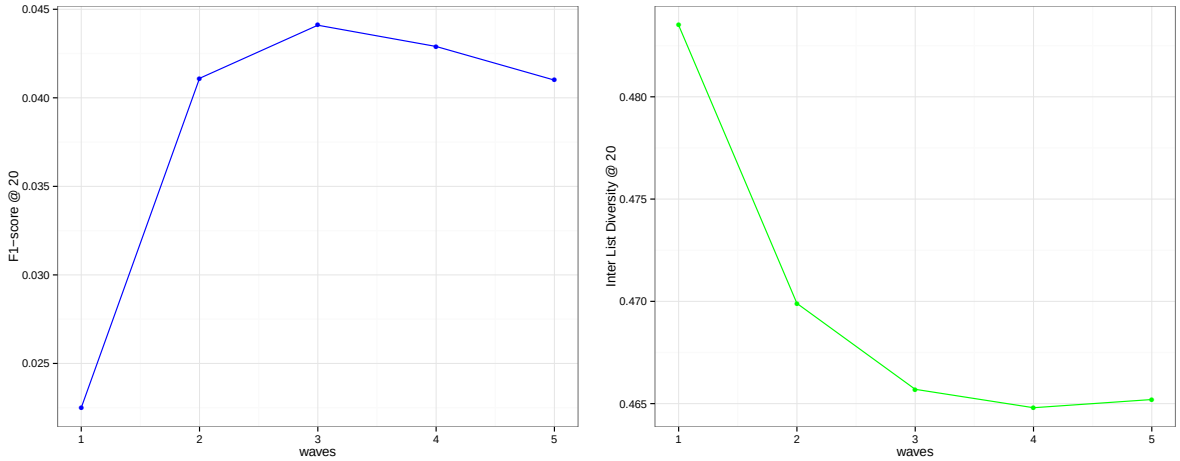

Figure 5.16.: Scores for F1@20 and ILD@20 for 1 to 5 waves, with 3 phases and activation threshold of 0.6.

After achieving good results with 4 phases, 3 waves and an activation threshold of 0.6, we tested different values for phases and waves, while keeping the activation threshold fixed at 0.6.

As the second step, we fixed the activation threshold at 0.6, based on the previous step. We fixed the phases at 3, and varied the waves from 1 to 5. The results are in Figure 5.16. We also generated results for 4 phases and variable waves from 1 to 5 in Figure 5.17, and for 5 phases and variable waves from 1 to 5 in Figure 5.18, both using the same activation threshold of 0.6.

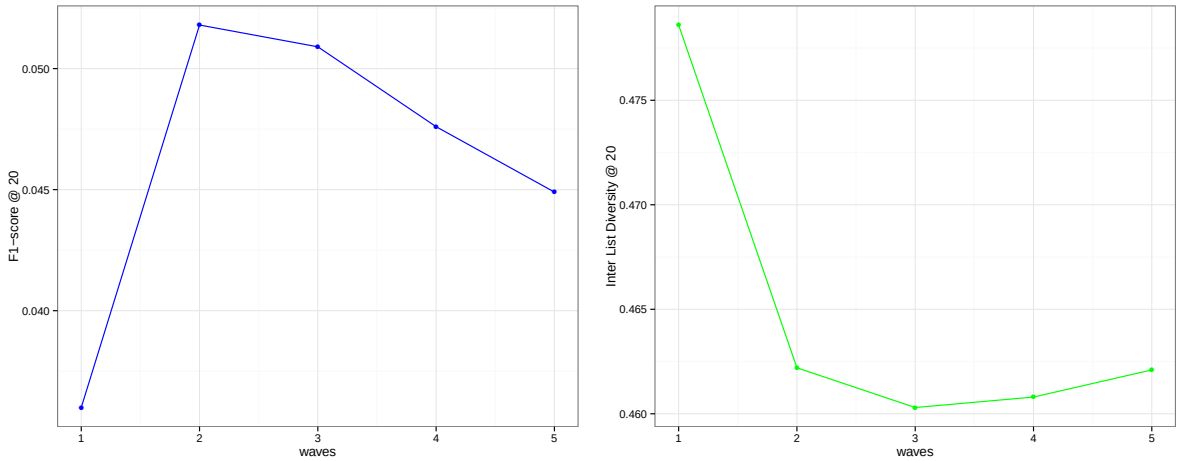

Figure 5.17.: Scores for F1@20 and ILD@20 for 1 to 5 waves, with 4 phases and activation threshold of 0.6.

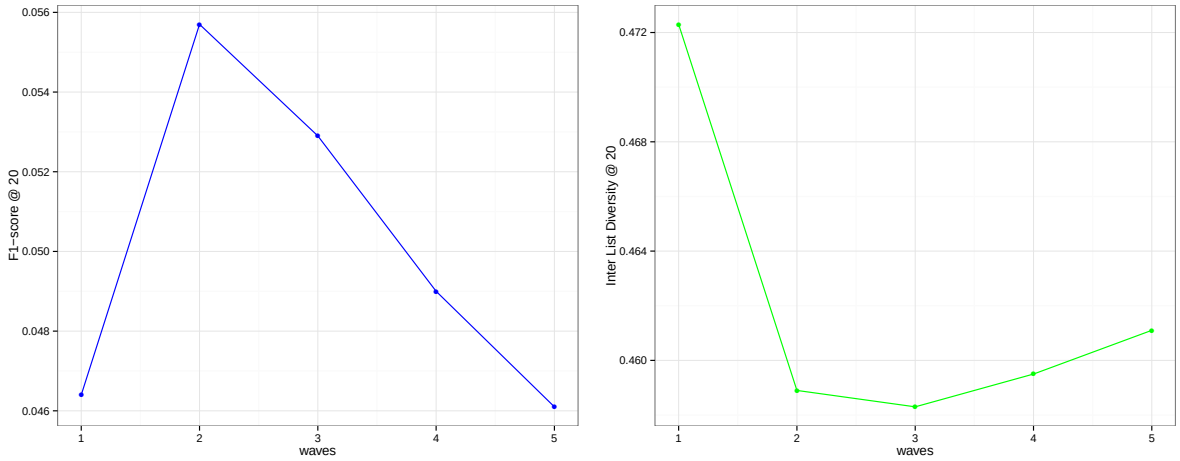

Figure 5.18.: Scores for F1@20 and ILD@20 for 1 to 5 waves, with 5 phases and activation threshold of 0.6.

The best ranking among our submitted results was achieved for 5 phases, 1 wave and an activation threshold of 0.6, with 0.0464 F1@20 and 0.4723 ILD@20.

Finally, as the third step, we tried to improve our ranking again, by fixing 5 phases and 1 wave, and testing values for the activation threshold from 0.3 to 1.0. The results are in Figure 5.19.

We achieved the best ranking among our submitted results for the activation threshold of 0.7, with 0.0413 F1@20 and 0.476 ILD@20.

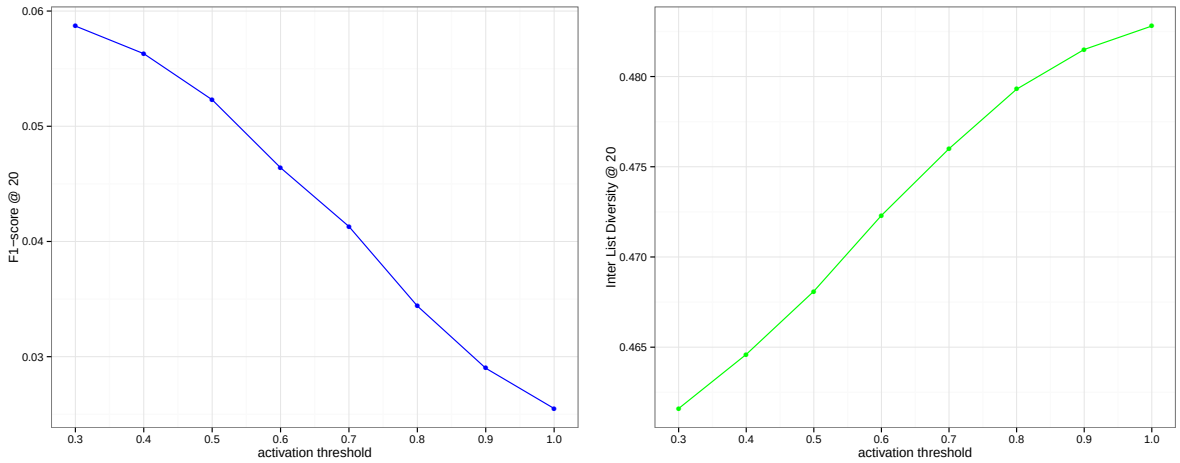

Figure 5.19.: Scores for F1@20 and ILD@20 for activation threshold values from 0.3 to 1.0, with 5 phases and 1 wave.

5.6.5. Discussion

In terms of the goals for participating at the LODRecSys challenge, we were able to show that SemStim has a competitive performance for the single-domain diversity recommendation task. In addition, we were able to show that the diversity of the results of SemStim is inversely correlated to the accuracy of the results.

The overall results for the diversity recommendation task at the LODRecSys challenge are in Figure 5.20. The results show that SemStim has a competitive performance which is comparable to that of the other participants in the diversity task. SemStim was placed on the 3rd place out of 12 participants in the leader board of the diversity recommendation task. This is based on the average of the rankings on F1-score and intra-list diversity (ILD). The F1-score of SemStim was ranked in 4th place with 0.0413, and the ILD of SemStim was ranked in 7th place with 0.476.

Further, we found that our algorithm is not over-optimised for either F1-score or diversity, as the parameters of the algorithm can be tuned towards either of these metrics. Independent of the algorithm parameters, the baseline of the ILD diversity is 0.4570. By setting the activation threshold very low at 0.1, and setting a maximum of 4 phases and 3 waves, SemStim can achieve an F1-score of 0.0593, which comes at the cost of a comparatively low ILD of 0.4591. On the other hand, SemStim can achieve a high ILD

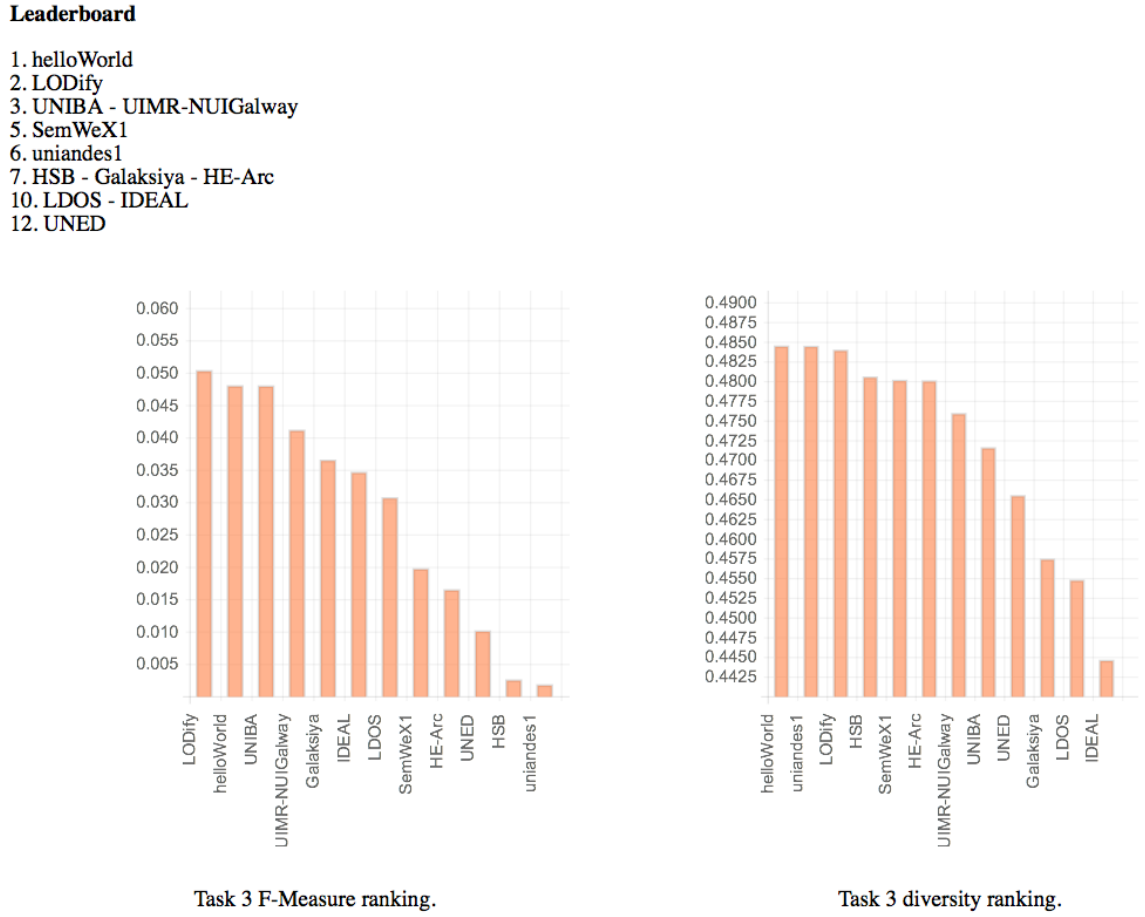

Figure 5.20.: Screen shot of the final ranking of participants for the diversity recommendation task. SemStim is identified by the team name "UIMR-NUIGalway". The screenshot is taken from the evaluation web site of the challenge.

of 0.4858, by setting a higher threshold of 0.6 and using only a maximum of 1 phase and 1 wave. However this is at the expense of a low F1 score of 0.0019.

This shows that the results of SemStim can be tuned towards having either a bias towards accuracy, or towards diversity or the results can be balanced.

5.7. Summary

In this chapter, we presented the results of evaluating our SemStim algorithm. Our evaluation consists of three different experiments. In addition, we describe the results of our participation in a recommender systems "implementation challenge".

We evaluated SemStim by comparing it to two other graph-based algorithms, Linked Data Semantic Distance (LDSD) and set-based breadth-first search (SetBFS). In addition, we compare SemStim to k-nearest neighbor collaborative filtering (CFknn), and SVD++,

which is a collaborative filtering algorithm based on matrix factorisation. Lastly we use random selection as worst case baseline.

For our evaluation, we first use the MovieLens 100k data to compare the single-domain accuracy of the algorithms. Then we demonstrate the capability of SemStim to generate cross-domain recommendations, with different source and target domains. We do this by using profiles with ratings in two different domains provided by Amazon as part of a cross-domain rating data set. In addition, we evaluate the diversity of the recommendations generated by the different algorithms in two different experiments.

The results of the single-domain accuracy experiment show that the performance of SemStim dominates the performance of the other algorithms for the top-k recommendation task with equal source and target domains.

The results of the cross-domain accuracy experiment show that SemStim is able to generate cross-domain recommendations. When using DVDs as the source domain and music as the target domain, SemStim outperforms the other algorithms with statistically significant results. However, when reversing the source and target domain, so that music is used as the source domain and DVDs as the target domain, SemStim performs only as good as SetBFS, with a not statistically significant difference between the performance of the two algorithms.

The different performance of SemStim for these two different pairings of the domains music and DVDs indicates that the performance of SemStim is dependent on the selection of the specific target and source domain. As future work, we plan to investigate the connection between the performance of SemStim and e.g. the size of a domain or the connectedness of the items in a domain.

An additional external factor for the performance of SemStim in this experiment, is the quality of the linkage data between the items in the Amazon data set and DBpedia. As we only used basic heuristics to create this linkage data, the performance of SemStim might be improved by using a more sophisticated entity resolution approach. While the topic of entity consolidation and linking of entities to DBpedia is beyond the scope of this thesis, it provides a promising direction for future research.

Further, we performed an experiment to compare the diversity of the SemStim recommendations with those of the other algorithms. For this experiment we again used MovieLens 100k rating data as both source and target domain. To determine the diversity of the recommendations, we use our own feature-based diversity metric, which is based on estimating the number of clusters in a set of recommendations. The results of this experiment show that the diversity of recommendations generated by SemStim can be tuned by changing the activation threshold of nodes, which is one of the parameters of the algorithm. However, tuning the recommendations this way, requires using all of the preferences of a user, including his negative and neutral user preferences. When only

using the positive preferences of a user, there is no correlation between the range of diversity and the activation threshold.

Then we describe the results of our participation in the "diversity recommendation task" at the Linked Open Data-enabled recommender systems challenge at the Extended Semantic Web Conference 2014. We were able to show that SemStim has a competitive performance (3rd place out of 12) for the single-domain diversity recommendation task. The challenge organisers used the average of the ranks achieved for F1-score@20 and intra-list diversity (ILD@20) to rate the participants. In addition, we were able to show that for recommendations on the data set of the challenge, the diversity of the results of SemStim is inversely correlated to the accuracy of the results.

In summary, our evaluation showed that SemStim is able to provide cross-domain recommendations, which do not require any overlap between target and source domains, and without using any ratings in the target domain.

5.7.1. Contributions of this chapter

In this chapter we have addressed research questions (Q1) and (Q4):

Research question (Q1): How can we mitigate the cold-start problem in order to provide recommendations for new users?

Research question (Q4): Which recommendation algorithm can generate cross-domain recommendations, without any overlap between source and target domain and without any ratings for the target domain?

Our evaluation of the SemStim algorithm has shown that, SemStim is able to provide cross-domain recommendations for domains without overlap. This in addition also shows how recommendations for a target domain can be generated from user preferences in a different source domain without any user preferences in the target domain. This shows that SemStim can address the cold-start problem for new users.

Chapter 6.

Use Case and Prototype Implementation*

In this chapter, we describe the prototype implementation, which is the outcome of the ADVANSSE[1] collaboration project with CISCO Galway. The goal of the ADVANSSE project was to provide a distributed platform for enabling personalised information access in an enterprise environment. The widespread use of social platforms in contemporary enterprises leads to the generation of large amounts of content shared through various social tools. The distributed nature of this information makes it difficult to fully exploit its value in an enterprise context.

While users of the Social Web have come to expect personalised services for various types of content such as music, books or social activity streams, personalisation for enterprise users remains a challenge. This is due to the distribution of users from the same enterprise amongst different social enterprise applications, such as blogs, wikis and micro-blogging applications, and the fact that the expertise of enterprise users can span multiple domains. We describe the ADVANSSE use case in more detail in Section 6.2.

Based on the ADVANSSE use case, we can derive three main requirements for the infrastructure and the algorithms that would allow the provision of social platforms in order to enable personalized information access in a distributed enterprise environment:

*This chapter is based on previously published work:

1. Benjamin Heitmann, Maciej Dabrowski, Conor Hayes, Keith Griffin, "Towards Near Real-Time Social Recommendations for the Enterprise", In Gloria Phillips-Wren, Liana Razmerita, Lakhmi C. Jain (ed.), *Innovations in Knowledge Management – The Impact of Social Media, Semantic Web and Cloud Computing*, Springer, Germany, 2014.

2. Benjamin Heitmann, Maciej Dabrowski, Conor Hayes, Alexandre Passant, Keith Griffin, "Personalisation of Social Web Services in the Enterprise using Spreading Activation for Multi-source, Cross-domain Recommendations", In *Proceedings of the AAAI Spring Symposium on Intelligent Web Services Meet Social Computing*, 2012.

[1] http://advansse.deri.ie/

1. User profiles are distributed across different information systems and social platforms in the IT landscape of the enterprise. This requires an architecture and a transport mechanism for aggregating such *multi-source user profiles* across the enterprise, before the distributed user preferences can be used for personalisation.

2. In the enterprise, user profiles represent the expertise of a user, which can span multiple domains, depending on the role of the user. As such, these *user profiles* are *domain-neutral*.

3. Finally, in order to leverage such multi-source and domain-neutral user preferences for personalisation, a *cross-domain recommendation algorithm* that can use preferences in one or more source domains to generated recommendations in a different target domain is required.

The requirements of the ADVANSSE use case match the requirements for personalisation in open ecosystems, as we have identified them in Chapter 2. This allows us to demonstrate the usage of our cross-domain personalisation approach in a real-world use case.

In addition to the architecture used for the ADVANSSE prototype, Appendix A presents a privacy-enabled architecture for open ecosystems. We describe how to combine existing infrastructure of the Web of Data and existing standards for decentralised identity management in order to achieve privacy-enabled user profile portability. The privacy-enabled architecture enables the creation of a universal private by default ecosystem with interoperability of user profile data. However, implementing our proposed privacy-enabled architecture was outside of the scope of this thesis.

Contributions of this chapter: In this chapter, we address the research questions (Q2) and (Q3). In addition we describe how we deployed the SemStim algorithm, and how we instantiate our conceptual architecture for personalised Linked Data applications for the ADVANSSE prototype.

1. (Q2) asks how we can support an open ecosystem for personalisation as an alternative to closed ecosystems. In Section 6.3, we describe the architecture and the technologies in the ADVANSSE prototype, which are used for transporting and aggregating user profile data, in order to enable an open ecosystem for personalisation.

2. (Q3) asks how domain-neutral user profiles can be implemented. In Section 6.4.1, we describe how we merge user profiles from multiple sources in the ADVANSSE prototype, and we describe the vocabularies used by the merged domain-neutral user profile.

3. The ADVANSSE prototype uses the SemStim algorithm, from Chapter 4, to make recommendations. We describe the details of our SemStim implementation and how we deploy the SemStim algorithm in the ADVANSSE prototype in Section 6.4.3.

4. The ADVANSSE architecture is based on the conceptual architecture for personalised Linked Data applications, which we introduced in Chapter 3. We describe how we instantiate the conceptual architecture in Section 6.3.5.

Chapter outline: The rest of the chapter is structured as follows: First, in Section 6.1, we introduce the background regarding distributed organisational environments. Then, in Section 6.2, we list the use cases of the ADVANSSE project. The use cases revolve around filtering of subscriptions, recommendations of posts and updating of interests. This is followed by the description of the architecture of the ADVANSSE distributed social platform in Section 6.3. We describe the protocols used for data synchronisation, as well as the different components employed by both the server and connected platform in the ADVANSSE architecture. Finally, we provide the details of the prototype implementation of the ADVANSSE distributed social platform in Section 6.4. First we describe the details of the data used for the prototype as well as the integration process, then we describe how we implemented the components of the ADVANSSE server and connected platform, as well as the implementation of the SemStim algorithm. We conclude the chapter by summarising the main contributions.

6.1. Background: Distributed Enterprise Environments

Personalised recommendations greatly enhance the user experience of searching, exploring and finding new and interesting content [Bobadilla et al., 2013], however, mostly using homogeneous data from one source. In contrast, the use of personalisation in a distributed enterprise context is more challenging, as the personalisation has to take multiple sources, formats and genres into account.

Organizations build and maintain many information systems to manage large volumes of various data published and consumed by knowledge-intensive workers [Gold et al., 2001]. With the shift to the Enterprise 2.0 [Passant, 2010c], the organizational landscape changes from a centralized environment with a small number of centralized content repositories to a more distributed model in which various peers can both publish and consume information.

This shift requires new approaches for delivery of timely and relevant information in a close-to-real-time manner across such peer-to-peer (P2P) [Oram, 2001] networks. Many initiatives [Weiss et al., 2007, Du & Brewer, 2008, Mukherjee et al., 2008] focus on building P2P wikis that combine the benefits of mass collaboration with the intrinsic qualities of peer-to-peer networks, such as scalability or fault-tolerance. One of the challenges in building such collaborative tools is to ensure the consistency of content replicated on different peers [Skaf et al., 2008].

On the other hand, knowledge workers in contemporary organizations utilize many tools that support collaboration through creation and sharing of information through corporate networks [Passant, 2010c]: blogs, wikis, or social networks. Nevertheless, many organizations report that crucial information is often not managed effectively [Sabherwal & Becerra-Fernandez, 2010], which affects efficiency and generates additional spending. To address this concern, organizations attempt to sustain information exchange through utilization of social tools both internally and externally. Once social connections are established, it is important to gather and reuse information available in this network.

The result of this shift from monolithic platforms to distributed enterprise platforms, is that suitable personalisation approaches need to take the distributed nature of user profiles with preferences from multiple sources and from multiple domains into account. Thus, the proposed ADVANSSE distributed social platform needs to provide support for an open ecosystem for personalisation, which allows addressing the two main challenges of personalisation in a distributed enterprise platform:

1. Aggregation and integration of user profile data from different information systems/social platforms in the IT landscape of the enterprise.
2. Providing personalisation by making use of multi-source and domain-neutral user profiles, which are aggregated by the underlying infrastructure.

As part of the ADVANSSE prototype we implement and deploy our proposed SemStim algorithm, which is introduced in Chapter 4, and which addresses the second challenge.

Addressing the first challenge of aggregating and integrating data from different social platforms in the enterprise, requires efficient data synchronization tools [Trams et al., 2011] to allow for timely updates and retrieval of relevant content. The generic categorization of models for communication and content/event exchange in a distributed environment differentiates pull and push approaches.

The *pull* model involves an initial request from the (active) client that is responded by a (passive) server and is one of the most commonly used communication patterns in distributed networks. *Polling* is a mechanism related to pull model, which relies on clients actively sampling the server status through repetitive requests. Polling is considered resource expensive and scales poorly [Eugster et al., 2000] as frequent polling may lead to inefficient usage of resources, but infrequent requests "may result in delayed responses to critical situations" [Eugster et al., 2000]. Further, many scenarios require asynchronous delivery of events for better performance and scalability. *Long Polling*, introduced to address these limitations, is an approach based on the request-response model in which the server holds the request open until the response is available (or when the set timeout is reached) [Griffin & Flanagan, 2010].

In contrast to the pull approach, the *push* model assumes a passive client that is notified of the occurrence of specific events upon a subscription to the server. The

publish/subscribe (PubSub) interaction paradigm exploits the push model as it enables agents to subscribe to a particular event (e.g. content update), and to receive asynchronous notifications from the server/publisher when the event occurs [Eugster et al., 2003]. The advantages of the PubSub paradigm over the Polling approach lie in the optimization of the number of request, the required network traffic, and in the full decoupling in "time, space, and synchronization between publishers and subscribers" [Eugster et al., 2003]. Although the push approaches are gaining more popularity, the tools built using the pull paradigm are prevalent (see [Bhide et al., 2002] for a detailed discussion). However, with the expansion of Semantic Web technologies, more focus is put on applications implementing the push interaction model.

For the proposed ADVANSSE distributed platform, we employ the publish/subscribe interaction model, as it provides a greater degree of decoupling between the distributed parts of the platform as compared to pull interaction. We make use of publish/subscribe interaction through the use of the XMPP protocol, which we describe in more detail later in this Chapter.

6.2. ADVANSSE use cases

In order to introduce the ADVANSSE architecture, we present three use cases. Each use case introduces new requirements for the overall architecture, for the XMPP infrastructure and for the personalisation approach. The first use case shows why filtering of messages is required in a social enterprise context. The second use case explains why adding of recommendations is beneficial for discovering content in an agile way, without requiring previous knowledge about people and their expertise. Finally, the third use case demonstrates the benefits of discovering existing expertise and existing content as soon as a user adds new interests to his profile.

6.2.1. Common setting for the use cases

Organisational boundaries: Andrew, Bob and Cecilia are knowledge workers, employed by the same organisation (the Umbrella corporation), however they work in different divisions of this organisation. Andrew is the head of the division A, which includes subdivisions B (IT department) and C (marketing department). Bob is an IT administrator, who works in subdivision B (the IT department). Cecilia works in marketing for subdivision C.

Expertise: In terms of expertise, Andrew, Bob, and Cecilia are all experts in Semantic Web technologies. In addition, Bob is an expert in the IT domain of databases, and Cecilia is an expert in the research domain of information retrieval.

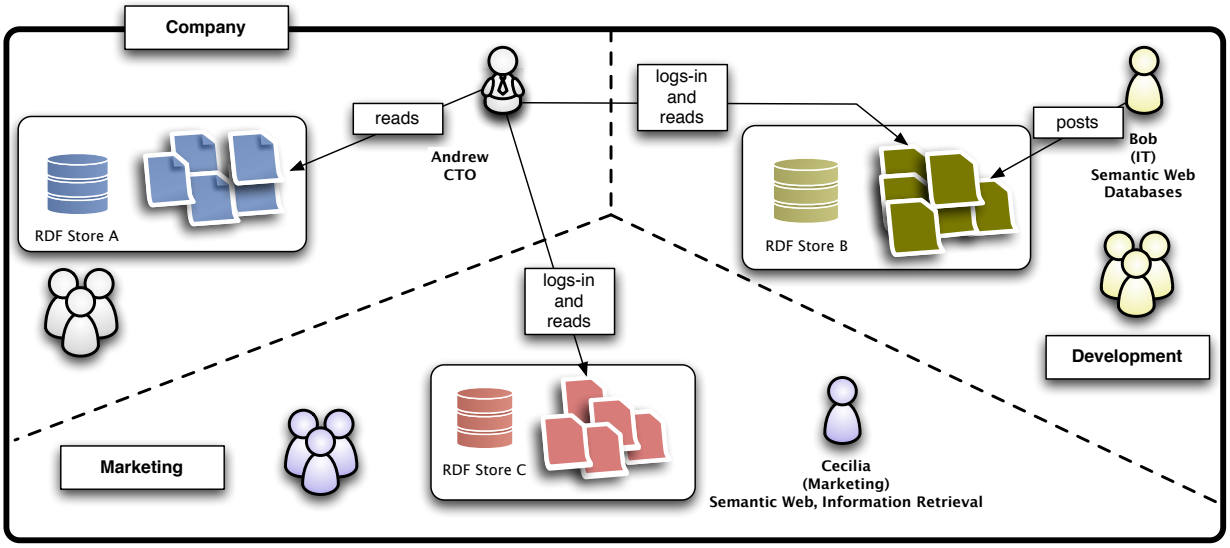

Figure 6.1.: Summary of the organisational boundaries between Andrew, Bob and Cecilia.

Common customer: All three are currently working with a common customer, the National University of Ireland, Galway (NUIG).

Figure 6.1 illustrates the organisational boundaries between Andrew, Bob and Cecilia, and Table 6.1 summarises the membership of the three co-workers in their respective subdivisions and their respective domains of expertise.

Employee	Member of subdivisions	Domains of expertise
Andrew	A (division head), B and C	Semantic Web
Bob	B (IT administrator)	Semantic Web, Databases
Cecilia	C (marketing)	Semantic Web, Information Retrieval

Table 6.1.: List of employers, subdivisions and domains of expertise

6.2.2. Use case 1: filtering of subscriptions

Cecilia works in marketing, and she would like to get updates on what her co-workers are doing. So Cecilia subscribes to both Andrew's and Bob's updates. Andrew is the division head, so she wants to follow all of his posts. Bob is working in the IT department, so Cecilia would like to only get his posts which are relevant from a marketing perspective. So Cecilia subscribes to Andrew's posts, but filters out all the posts which are not tagged with "marketing".

6.2.3. Use case 2: recommendations of posts

Being aware of the overwhelming amount of content being published daily within the organisation, Andrew would like to be notified when any new post related to his domain of expertise, the Semantic Web, is published in subdivisions B or C. He would like to get notifications for any new and relevant topic in his social dashboard on platform A, without using the social platforms used by subdivisions B or C. In addition, he would like to be notified of relevant content even if it is published by co-workers to which he is not subscribed in either platform B or C. So if Andrew does not know of Bob's presence in platform B, Andrew would still like to get notifications about recommended content from Bob.

6.2.4. Use case 3: Updating of interests and recommendations

Umbrella corporation is developing a new database product called "ProjectX". As division head, Andrew is part of the development team of the new product, and he publishes his posts about the product with the tag "ProjectX". Now Cecilia joins the development team in order to start creating some marketing drafts. So she adds the "ProjectX" tag to the interests in her user profile in order to receive recommended posts about the new project. She also adds the tag "database" in order to read up on databases in general, so that she can better target the marketing message. However, that exact tag was not used for any post before. Instead her co-workers used tags like "RDBMS" and "information retrieval". Cecilia expects the system to recommend posts using such similar tags to her in addition to direct matches for the "database" tag.

6.3. The ADVANSSE Distributed Social Platform

Organizations build and maintain many information systems to manage a large volume of content published and consumed by knowledge-intensive workers [Gold et al., 2001]. These systems are inter-connected both internally, within an organization, and with systems external to the organization, for example news feeds on the Web[2]. Such environments involve many peers, which share information within a large network that is often distributed across various departments, or sometimes even geographically.

In order to provide a personalisation approach for such distributed social platforms, we need to provide an architecture which is capable of aggregating and integration data from distributed social platforms.

[2]see for example http://www.reuters.com/

In this section we describe the architecture of our proposed ADVANSSE distributed social platform. We first introduce the protocols used to synchronise data between different, distributed social platforms. Then we describe the different components of the ADVANSSE architecture.

6.3.1. Protocols Used for Data Synchronisation

For some applications (e.g. news feeds), content updates can be represented as streams of RDF triples [Shinavier, 2010] and the problem of RDF content integration can be viewed as a succession of RDF documents available through a feed. Many applications in the Enterprise 2.0 are more stateful (e.g. presence management), thus require not only addition of the new content but also deletion/editing of existing information. Thus, RDF data integration techniques deployed in Enteprise 2.0 platforms should support not only addition, but also deletion and editing of existing content. Further, it is essential that the update operations should be done on the lowest possible level, which is the level of individual RDF triples.

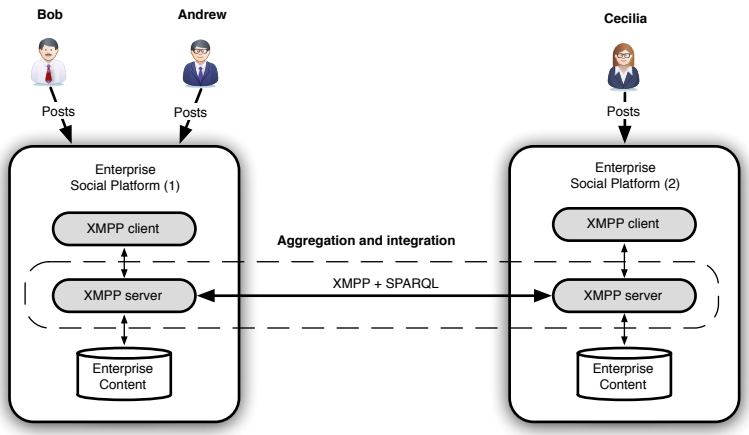

Figure 6.2.: Aggregation of content in the ADVANSSE architecture using XMPP and SPARQL Update.

For the ADVANSSE distributed social platform, we propose to use the XMPP protocol for transporting data between the server and the connected platforms, while using the SPARQL update protocol to describe the data inside of the XMPP messages. An overview of how XMPP and SPARQL update are used for synchronisation and aggregation of content in the ADVANSSE architecture is given in Figure 6.2.

SPARQL update

SPARQL Update[3] provides a standard update language for RDF graphs. SPARQL Update is a part of SPARQL 1.1, which has reached the status of a W3C recommendation in August 2013. With the syntax derived from SPARQL, SPARQL-Update provides operations to create and remove graphs, as well as to update existing graphs and RDF triples, including *INSERT*, *DELETE*, *CLEAR* or *LOAD*. The granularity of these commands enables triple-level operations on a given RDF graph.

The Extensible Messaging and Presence Protocol

XMPP is an open technology for real-time communication, which enables a wide range of applications including instant messaging, online presence, lightweight middleware, content syndication, and generalized routing of XML data. The core technology behind XMPP was standardized in the XMPP RFCs in 2004, revised in 2011[4]. Although not coupled with any specific network architecture, XMPP usually has been implemented over TCP used for client-server and server-server connectivity. Most clients connect directly to a server over a TCP connection and use XMPP to take full advantage of the functionality provided by a server and any associated services. Further, the XMPP PubSub extension, known as XEP-0060[5] offers an implementation of a pubsub paradigm. The recovery mechanism provided by the XMPP is an important feature that increases reliability of change distribution. When a given server (A) attempts to notify another server (B) about RDF updates to apply and the connection between the two servers is lost for any reason, the protocol implementation provides a mechanism that caters for reconnecting the servers.

Using XMPP and SPARQL Update together

The XMPP server has the function of routing and transporting messages between the social platforms which are connected to it. In addition it provides an extensible platform that can process all information that it receives, for example to provide content personalisation through pluggable personalization components. As one or more platforms may be connected to a given XMPP server, the servers communicate with each other sending SPARQL Update messages embedded in XMPP PubSub stanzas. XMPP provides the infrastructure for connecting social platforms in a decentralised way and for sharing of knowledge between those social platforms. The server provides a central point for the connected social platforms to exchange XMPP messages. In case an existing XMPP

[3] http://www.w3.org/TR/sparql11-update/
[4] http://xmpp.org/rfcs/rfc6120.html, http://xmpp.org/rfcs/rfc6121.html, http://xmpp.org/rfcs/rfc6122.html
[5] http://xmpp.org/extensions/xep-0060.html

server is used, it must implement the XMPP Publish-Subscribe extension (XEP-0060)[6], which allows XMPP clients to subscribe to updates. In order to publish and receive aggregated content social platforms need to connect to the XMPP server via the XMPP client. This can be accomplished using an open source component[7] that allows easy integration.

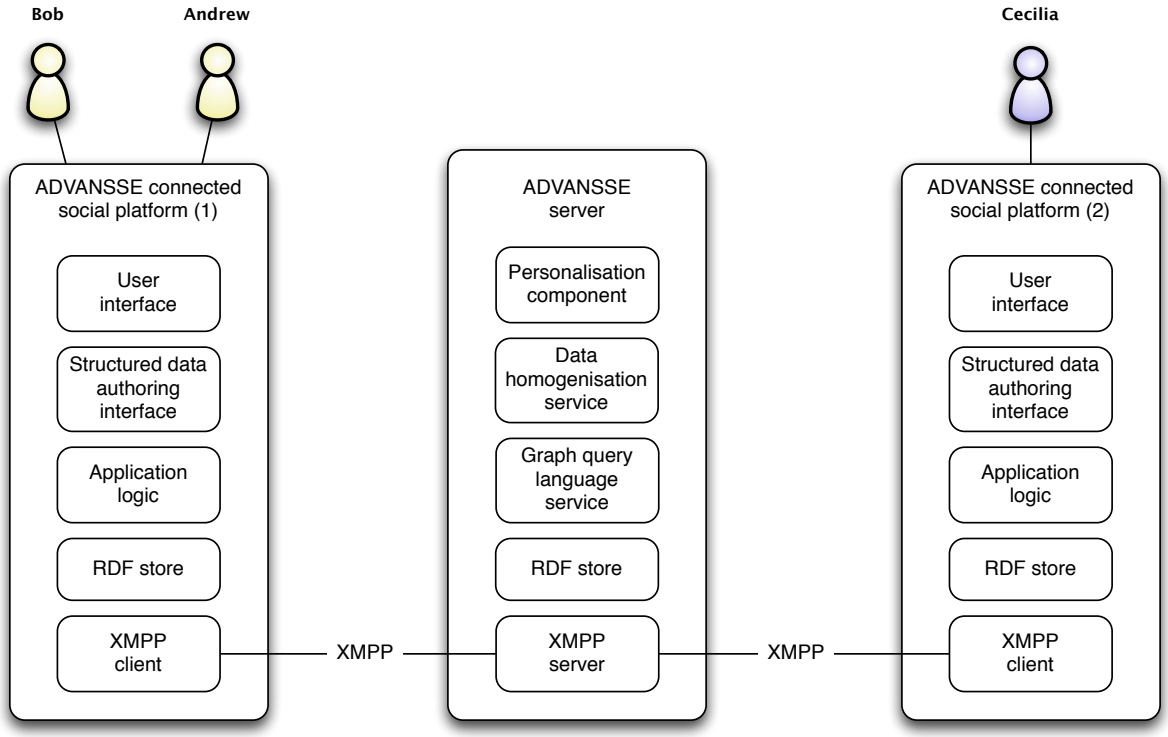

Figure 6.3.: High level overview of the ADVANSSE architecture

6.3.2. ADVANSSE Architecture

We now describe the ADVANSSE architecture which uses XMPP and SPARQL Update to allow different, distributed social platforms to connect to each other via a central ADVANSSE server.

Figure 6.3 presents a high level overview of the ADVANSSE architecture. The ADVANSSE server implements all the functionality which can be centralised, in particular it allows for routing and transporting of XMPP messages between all connected platforms and for personalisation using the meta-data of transported content. The ADVANSSE connected social platforms implement the user-facing systems in the distributed platform.

We describe the ADVANSSE server and the ADVANSSE connected social platforms in more detail in the following Sections. We then discuss the scalability of the ADVANSSE

[6] http://xmpp.org/extensions/xep-0060.html
[7] https://github.com/derixmpppubsub/derixmpppubsub/

architecture, and describe how we instantiate the conceptual architecture for personalised Linked Data applications.

6.3.3. Components of the ADVANSSE Server

The ADVANSSE server has the function of routing and transporting messages between the social platforms which are connected to it, so it contains an *XMPP server* component. In addition it provides the service of personalisation of content, for which it requires a *personalisation component* and an *RDF store* for persisting content meta-data and user profiles. In addition, it contains a *graph query language service*, which allows querying the RDF store. Finally, as different connected social platforms could be using different namespaces for their tags, it contains a *data homogenisation service*.

XMPP server: In the ADVANSSE architecture, XMPP provides the infrastructure for connecting social platforms in a decentralised way and for sharing of knowledge between those social platforms. The ADVANSSE server provides a central point for the connected social platforms to exchange XMPP messages. The XMPP server component implements the XMPP Publish-Subscribe extension (XEP-0060)[8], which allows XMPP clients to subscribe to updates, which are published to a specific node on the XMPP server by a publisher.

Personalisation component: The personalisation component provides users with recommendations for content which is relevant to their interests. It has access to the meta-data about user generated content and to the user profiles. It then executes the recommendation algorithm on top of the content meta-data and the user profiles, after which it uses the XMPP server component in order to deliver the recommendations to the users. The recommendation algorithm is executed in order to generate new recommendations. It uses the user profiles and the content meta-data as input. In addition it uses other sources of background knowledge. The result is a ranked list of recommendations for different types of items (e.g. other users, posts and tags) for every user.

RDF store: The XMPP server receives meta-data about newly published content from the XMPP clients which are connected to the server. This meta-data is contained in SPARQL 1.1 Update operations which are transmitted as XMPP IQ messages. The XMPP server stores this meta-data to an RDF store, so that it can be accessed by the personalisation component.

Graph query language service: Provides access to the RDF store via a SPARQL endpoint.

[8] http://xmpp.org/extensions/xep-0060.html

Data homogenisation service: This component is responsible for identifying similar entities used in different namespaces or in different data sets. One of the most important tasks of the data homogenisation service is to find concepts from the background data of the recommender system which match tags used in the user profiles and the content meta-data.

6.3.4. Components of the ADVANSSE Connected Social Platform

Multiple social platforms can be connected to the same ADVANSSE server. These social platforms allow users to create new content, and to search existing content or to view recommendations of content. In order to connect to the ADVANSSE server an XMPP client component is required. The social platform provides an end-user facing user interface through a web application. Finally, it has its own RDF store for persisting content and user profiles.

XMPP client: In order to participate in the ADVANSSE architecture, social platforms connect to the ADVANSSE server via the XMPP protocol and the Publish-Subscribe XMPP extension (XEP-0060).

User interface: In order to provide a user interface for the users of the social platform a user interface is provided.

Structured data authoring interface: In addition to the user interface, which provides general user interface elements, each connected social platform provides an authoring interface for structured data. This component allows authoring of new content, and uses the application logic component to check constraints and rules when creating new content.

Application logic: This component implements the specific application logic of the connected social platform. It uses the content store for persistence of user profiles, and it uses the XMPP client component to send the content to the ADVANSSE server.

RDF store: Each social platform in ADVANSSE stores the user profiles and the content generated by its users in persistent storage, so that it can be retrieved at a later time again.

6.3.5. Instantiating the Conceptual Architecture for Personalised Linked Data Applications

The ADVANSSE architecture is an instantiation of the conceptual architecture for personalised Linked Data applications, which we introduced in Chapter 3.

We use most of the components of the conceptual architecture. In particular, in the previous sections we have described how the ADVANSSE architecture makes use of the following components:

- RDF Store
- Graph Query Language Service
- Data Homogenisation Service
- Personalisation component
- Application logic
- User Interface
- Structured Data Authoring Interface

The main difference between the conceptual architecture and the proposed ADVANSSE architecture, is the distributed nature of the ADVANSSE architecture. Instead of putting all the components into one application, the components are distributed between the ADVANSSE server and the ADVANSSE connected social platforms. However, the XMPP protocol provides the connection between the server and the social platforms, so that the server can provide a personalisation service using data from the connected social platforms.

6.3.6. Scalability of the ADVANSSE architecture

The proposed architecture allows scaling the different parts of the architecture to accommodate growth in the number of connected social platforms and the number of servers:

Scalability of servers: The XMPP protocol is fully decentralised, as it does not require any central server with knowledge about the whole network. Instead routing and transport of XMPP messages happens in a decentralised way. This makes it possible to have multiple ADVANSSE servers. In this scenario each server manages the messages of the social platforms which are connected to that server. In addition, all servers are connected to each other, so that the users of any social platform which is connected to one server can send messages to users of any other social platform connected to any other server.

Scalability of personalisation: The personalisation component is implemented centrally on each ADVANSSE server, as the personalisation needs to be able to use the meta-data of all the messages passing through the server. However, when there are multiple servers, it is sufficient to subscribe each server to the XMPP PubSub

updates of all other servers in order to allow personalisation on multiple servers at the same time.

Scalability of connected social platforms: Each server can be connected to multiple social platforms due to the choice of using the XMPP protocol to connect servers and social platforms.

6.4. Prototype Implementation

In order to provide a prototype of the ADVANSSE architecture, we have implemented a prototype demonstrator. In this section we will describe the implementation of all components, as well as the details of the data sources and data schemata used for the prototype.

We show how we implement domain-neutral user profiles in Section 6.4.1, as part of our description of the data used in the prototype. And we show how we deploy the SemStim algorithm for the ADVANSSE prototype in Section 6.4.3, as part of our description of the implementation of the prototype.

6.4.1. Prototype Data

For demonstration purposes we use data which is as similar as possible to the data which is described in the use cases. Data about user profiles, tags and posts is provided by StackExchange, semantic background knowledge is provided by DBpedia, and the Cisco ERT vocabulary is used as the schema for representing and integrating all data into RDF. We describe all three kinds of data in the following sections.

User profiles, tags and posts from StackExchange

The use cases require data about users and their interests, as well as posts authored by the users. In addition, both user interests and posts need to use the same set of tags (the same namespace for tags). The closest available source of real-world data to approximate this kind of data is provided by the StackExchange network of question answering sites.

StackExchange[9] operates a network of many sites where users can post a question. Then other users try to answer the question, and the user who asked the question marks one answer as being the correct answer. In order to encourage a productive interaction between all participants, correct behaviour such as posting of descriptive questions, tagging of questions, and correct answers are rewarded with a point system. The most

[9] http://stackexchange.com/

prominent site of the StackExchange network is StackOverflow, which was also the first site. StackOverflow[10] is a generic IT answering site, however there are sites for a wide range of domains, from system administration to cooking and photography. Any user who has an account on any StackExchange site, can use the same account to post on any other site of the network. This also makes StackExchange one of the few sources of cross-domain user profiles, as the data dump allows reconstructing user accounts from their fragments on multiple sites.

StackExchange makes all data from all sites available in an anonymised data dump. The data dump is under a creative commons licence (Attribution-ShareAlike 2.5 Generic (CC BY-SA 2.5))[11] which allows commercial use of the data. The data is made available as a Microsoft SQL Server XML dump [12]. We used the data dump from September 2011.

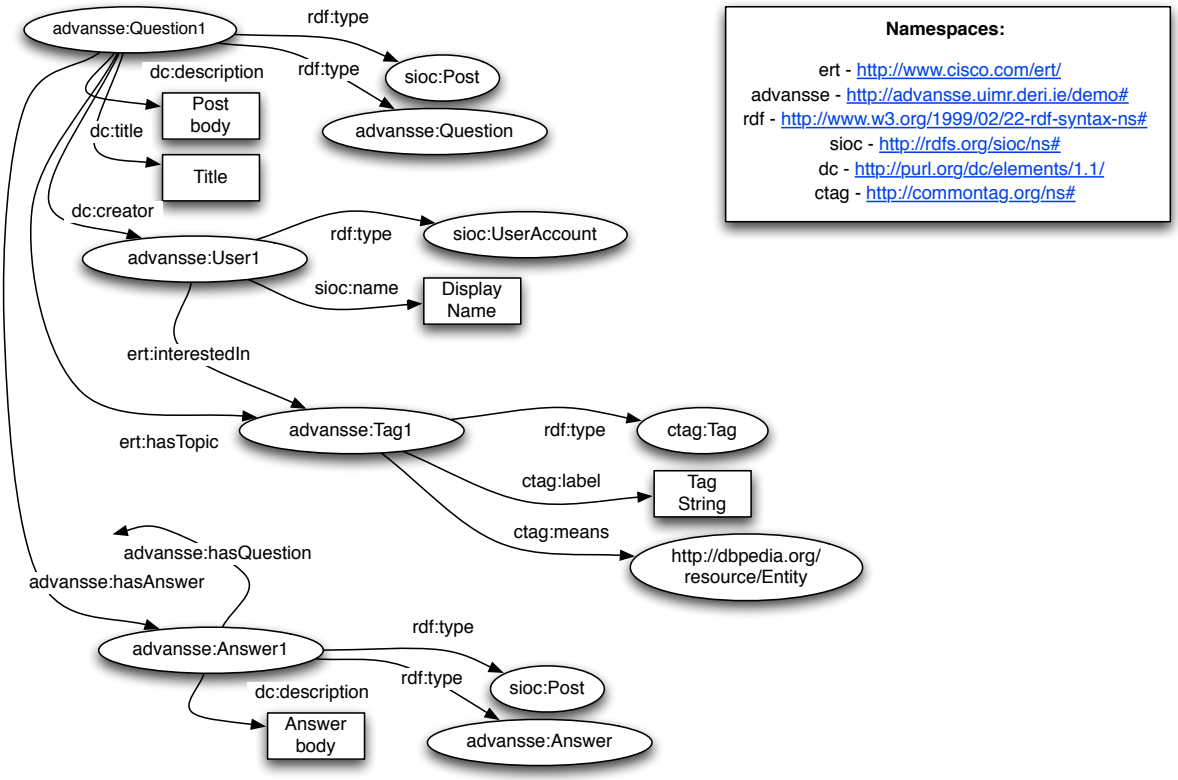

Figure 6.4.: Schema of the transformed StackExchange data

Data schema: CISCO ERT

For transforming and converting the StackExchange data into RDF we use the Cisco ERT Schema version 10. In addition, we added properties as necessary for the demonstration. Figure 6.4 shows the schema.

[10] http://stackoverflow.com/
[11] https://creativecommons.org/licenses/by-sa/2.5/
[12] http://www.clearbits.net/creators/146-stack-exchange-data-dump

Posts have the `rdf:type sioc:Post`. StackExchange makes a distinction between questions and answers. Questions have an additional `rdf:type advansse:Question`, in addition they have a `dc:description` string with the post body, and a `dc:title` string with the post title. Questions can be tagged via the `ert:hasTopic` property, while answers are not tagged. Answers have an additional `rdf:type advansse:Answer`, they have a `dc:description` string, but no `dc:title` string. Questions are connected to answers via `advansse:hasAnswer`, and via `advansse:hasQuestion` vice versa.

Users have the `rdf:type sioc:UserAccount` and they have a `sioc:name` string containing the display name on StackExchange. In addition a user can be connected to zero or more tags via the `ert:interestedIn` property.

Tags have the `rdf:type ctag:Tag`. The string of the tag is expressed in the `ctag:label` string. A tag can be connected with another URI of the same meaning via the `ctag:means` property.

StackExchange data extraction process

In order to use the StackExchange data for the demonstrator, we extracted and transformed the data in multiple steps:

1. Import of XML data dump into MySQL relational database: The StackExchange data dump contains an XML file per site. The XML file has been exported from Microsoft SQL Server 2008. A python script reads the XML file for each site, and then writes the data into a MySQL relational database.

2. Selection of sites: After inspecting the converted data in the MySQL database, a decision was made to select three sites and use their data for the demo: IT Security[13], Web applications[14] and Bicycles[15]. The sites where chosen with the following criteria: The IT Security and the Web applications site have content which is very close to the content which might be created in an enterprise knowledge sharing setting. The Bicycles site community has a significant overlap with the two other sites, so it was added to realistically simulate adding of expertise from a non-related domain to the user profiles.

3. Extraction of relational data into an RDF graph: A second python script then accesses the MySQL relational database, and queries the data which is required for the demonstrator, and creates an RDF graph using the schema described in Section 6.4.1. Only data from users which had accounts on all three sites where extracted.

[13] http://security.stackexchange.com/
[14] http://webapps.stackexchange.com/
[15] http://bicycles.stackexchange.com/

The result is an NTriples file which contains the graph with all interests, posts and tags from all users which have used all three sites.

The resulting data set contains 371 users who have at least one interest or one post in one of the three sites. These users have authored a total of 752 questions and 496 answers (not every question has a valid answer). In addition there are 607 tags in the graph. The StackExchange graph has a total of 2200 entities and 15000 edges.

To enable the SemStim algorithm to use the DBpedia data together with the StackExchange data, links between similar entities in both data sets need to be discovered. This is done by the data homogenisation service, which we describe in Section 6.4.2.

Background knowledge from DBpedia

The background knowledge which provides connections between related concepts is provided by DBpedia. We used a subset of DBpedia version 3.7[16], which was available during the development time of the demonstrator. In particular we used the following files:

- `mappingbased_properties_en.nt` Entities and their properties, extracted from Wikipedia info-boxes using strict ontology-based extraction by the DBpedia project.
- `article_categories_en.nt` Connections between entities and categories, extracted from the categories of pages on Wikipedia.
- `skos_categories_en.nt` Connections between categories themselves, extracted from the Wikipedia category tree. Encoded using the SKOS vocabulary, e.g. broader or narrower connections between two categories.
- `disambiguations_en.nt` Disambiguation links between DBpedia entities.
- `redirects_en.nt` Redirects between DBpedia entities.

The resulting DBpedia 3.7 subset includes all available links between entities themselves. As the spreading activation algorithm takes only links into account, while ignoring strings, this was the most important consideration when choosing the subset. The resulting data has a size of 5.46 GB in raw NTriples. The resulting graph contains 11 million entities and 40 million edges.

[16] http://wiki.dbpedia.org/Downloads37

6.4.2. Implementation of the ADVANSSE Server

We now describe the implementation of the different components of the ADVANSSE server prototype, and explain the design decisions for each component of the prototype implementation in the next sections.

XMPP Server: Openfire

We are using a standard installation of Ignite OpenFire[17] version 3.7.0 as XMPP server, which is configured for the demo with a set of user accounts, and some initial subscriptions. The server distributes the XMPP stanzas it receives from active publishers to relevant subscribers. RDF content is distributed in the form of SPARQL 1.1 Update queries embedded in XMPP IQ stanzas.

In order to implement all capabilities which are required for the ADVANSSE server, the XMPP server component needs to be integrated with an RDF meta-data store and with the personalisation component. Ignite OpenFire can be extended with plug-ins, however the extension mechanism is very restrictive. Plug-ins can only be executed when certain events are occurring, and the way in which events can be specified is not very expressive. As such, we were not able to use the official OpenFire extension points to integrate the RDF meta-data store and the personalisation component into Ignite Openfire.

However, we choose a different design which allowed the RDF meta-data store and the personalisation component to receive and react to XMPP messages: We integrated the RDF meta-data store into the personalisation component, and we added an XMPP client to the personalisation component. This allows us to specify the interaction between the RDF meta-data store, the personalisation component and the connected social platforms programmatically in terms of XMPP API methods and XMPP subscriptions. We used the Ignite Smack[18] java library as an XMPP client to implement this.

Personalisation Component

The personalisation component provides the integration point for the recommendation algorithm, the RDF meta-data store and the XMPP server, as the XMPP server itself is not extensible enough to add components with the functionality which ADVANSSE requires. In order to recommend posts to users or to get updates of user interests, the recommendation algorithm needs to save this data in persistent storage. In addition, recommendation results need to be stored persistently between executions of the recom-

[17] http://www.igniterealtime.org/projects/openfire/
[18] http://www.igniterealtime.org/projects/smack/

mendation algorithm. This allows the personalisation component to manage the delta between old and new recommendations.

The Personalisation component uses the Ignite Smack[19] XMPP client library in order to subscribe to all users on the connected social platforms. New posts and updates to user preferences are then stored in the RDF meta-data store and forwarded to the recommendation algorithm. The personalisation component uses SPARQL over HTTP to access the RDF meta-data store, and it uses a combination of SPARQL over HTTP and Java API calls to access the recommendation algorithm.

RDF Storage: HDT and Apache Jena

The ADVANSSE server uses RDF to store persistent meta-data about all published posts, and for determining new and already seen recommendations. In addition, the data about user preferences is stored as RDF.

However, the ADVANSSE server uses two RDF stores at the same time in order to store all the RDF data which it requires. The reason for this, is that the SemStim implementation requires a very scalable and fast RDF store. However, at the time of writing, the best candidate for this requirement is the read-only Header-Dictionary-Triples (HDT) RDF store [Fernández et al., 2013]. Therefore Apache Jena[20], which provides a read-write RDF store was used in conjunction with HDT.

Both RDF stores have performance characteristics and feature sets which complement each other:

HDT is optimised for fast querying and for handling of large data sets completely in-memory. HDT uses a compressed binary tree for storage, which allows it to fit large data sets entirely in memory and answer queries very fast. The DBpedia graph used for the demonstrator has 11 million entities and 40 million edges. Using HDT for this data set results in a compression factor of 92%, as the original size of the NTriples is 5.46 GB and the HDT file is 436 MB. The average duration of a simple subject or object query on this data set is 0.2 milliseconds. However, the **drawback** of HDT is that it is a read-only store. HDT is used to hold all data from the DBpedia data set, as this is static data.

Apache Jena TDB provides the read/write capabilities which complement the read-only, high-performance HDT store. Jena TDB is a persistent RDF store from the Apache Jena project. Jena TDB is used to hold all data from the StackExchange data set, as new user profiles and new posts need to be added to this data set dynamically. The StackExchange data set consists of 2200 entities and 15000 edges, which is 0.003% of the DBpedia data set. In addition, Apache Jena provides a SPARQL

[19] http://www.igniterealtime.org/projects/smack/
[20] https://jena.apache.org/

query endpoint, in the form of Jena Fuseki[21]. Jena Fuseki wraps around Jena TDB by providing an HTTP endpoint for SPARQL queries and SPARQL updates. This allows the full range of insert, update, delete and query operations.

Using Jena TDB in conjunction with HDT for the spreading activation algorithm results in a factor 2 increase in execution time, when compared to pure usage of HDT. However, this is preferable to the speed of using only Jena TDB for all data, which would result in long delays when executing the spreading activation algorithm. Using both stores together allows us to combine fast query execution of HDT on 99% of the data with data modifications on 1% of the data.

Data Homogenisation Service

In order to use the extracted StackExchange data set together with the background knowledge from DBpedia, it is necessary to link the tags from StackExchange to concepts from DBpedia. StackExchange tags are represented by simple strings, while DBpedia concepts are identified by a URI.

For the ADVANSSE demonstrator, we implemented a base-line link resolver as a python script. Links get resolved from a plain-text tag to a DBpedia URI in three steps:

1. Assign URIs to tags: Each tag used in the StackExchange data set, is assigned a URI in the `advansse` namespace, and the URI is given the `rdf:type ctag:Tag`
2. Match tag string to Wikipedia page URI: The tag string is resolved to one or more Wikipedia page URIs, by using two Wikipedia search engines: (a) the Wikipedia Opensearch API[22] which does fuzzy matching and which returns exactly one (the highest ranked) result; and (b) the Wikipedia full text Query API[23] which returns the top ranked 5 results. If the Opensearch API returns a result, we use that result, otherwise we use the first result from the full text query API.
3. Match Wikipedia page URI to DBpedia entity URI: After obtaining the most likely matches on Wikipedia, we use the DBpedia SPARQL endpoint to find the DBpedia entity URI which corresponds to the Wikipedia page URI. DBpedia encodes this connection with the `foaf:primaryTopic` property.

The result is a set of RDF statements, which connects tag URIs with dbpedia URIs via the `ctag:means` property. These RDF statements are loaded into the RDF store together with the StackExchange and DBpedia data sets.

[21] https://jena.apache.org/documentation/serving_data/index.html
[22] https://www.mediawiki.org/wiki/API:Opensearch
[23] https://www.mediawiki.org/wiki/API:Search

6.4.3. Recommendation Algorithm

The ADVANSSE prototype uses the SemStim algorithm, from Chapter 4, to make recommendations. We now describe how we implemented and deployed the SemStim algorithm for the ADVANSSE prototype.

The demonstrator implementation uses the following input and background data to provide its recommendation output:

Input data: User profiles and posts from the **StackExchange** data set, which also provides a set of plain-text tags used for both the user profile interests and post tags.

Background knowledge: The **DBpedia** graph together with the results of the **link resolver** provide background information linking similar tags from the StackExchange data indirectly via DBpedia.

Recommendation output: The recommendations for one user are three ranked lists of the **top-k posts, users and tags** respectively.

The recommendation algorithm is triggered by the personalisation component via a Java API. In addition all RDF updates with new posts and user interest updates, are passed from the personalisation component to the recommendation algorithm, which uses them to keep its own data up-to-date.

Benchmark results:

In order to test the performance and the success-rate of the implementation, we use the StackExchange data set to run a benchmark. The data set contains 371 users which have an average of 6 interests. Most user nodes have a degree between 2 and 5, with a long tail up to 51, and two outliers who have a degree of 140 and 176.

We experimented with different configurations for running the algorithm on the StackExchange data set, and the configuration with the best results (the highest number of users with the required number of recommendations for posts, tags and users) is the following: activation threshold: $\tau = 0.5$; maximum number of waves: $w_{max} = 10$; maximum number of phases: $\rho_{max} = 1$; default weight of predicates: $\beta = 1.0$; initial node output before applying modifiers: $\alpha = 4.0$. We set $\theta = 10$ as the required number of activated nodes from the target domain.

We run the algorithm three times, once for each of the three different target domains (1) posts, (2) users and (3) tags. When the algorithm terminates, we rank the activated nodes by their activation value and return the ranked list.

The target of 10 recommendations each for users, posts and tags could be reached for 315 of the 371 users, which is a success rate of 85%. Out of the 56 users without enough recommendations, 64% had only 2 out-links and 20% had only 3 out-links (These out links are interests and/or authorship links.) This indicates that interests or posts which are not very popular represent a problem when generating recommendations, resulting in a form of cold start problem for users with not enough interests and authored posts.

The performance of the implementation allows on-demand calculation of recommendations. The most atomic operation of the algorithm is a simple query on the HDT store. Such a simple query provides a subject or an object and two wildcards. The average length of such a simple HDT query is 0.2 milliseconds, however the duration increases linear with the number of results. This results in an average duration of 200 milliseconds for one run of the spreading activation algorithm. The average case scenario then is to generate recommendations for 5 users per second, or for one user on-demand (in less then a second).

6.4.4. Implementation of an Example Connected Social Platform

The ADVANSSE architecture allows multiple social platforms to connect to the ADVANSSE server via the XMPP protocol. For the demonstrator we have implemented a simple web site which allows users to emulate the behaviour on a social platform. Our web site persists the user generated content and the user preferences to local RDF storage, and connects to the ADVANSSE server via an XMPP client. Figure 6.5 shows a screenshot of the user interface of the prototype connected social platform. We implemented the prototype social platform as a Java servlet using Apache Tomcat[24] 7.0.

User interface and structured data authoring interface: In order to emulate the behaviour of users on a social platform, we have implemented a user interface. It allows users to create accounts and log into the site itself. On the site users can create posts with tags, add and remove tags from their list of interests, subscribe and unsubscribe from other users, add keyword filters to subscriptions. In addition users can view recommendations for interests, users and posts. The screenshot in Figure 6.5 shows the user interface. The upper part shows a user with his interests. The area labelled “extended profile” contains recommended tags for the user. The lower part of the screen shot shows the recommended content for the user. On the left-hand side are the posts which have been created by subscribed users. The middle part shows recommended posts which have been created by users to which the active user is not subscribed. On the right-hand side are users who are recommended as they have similar interests.

[24] https://tomcat.apache.org/

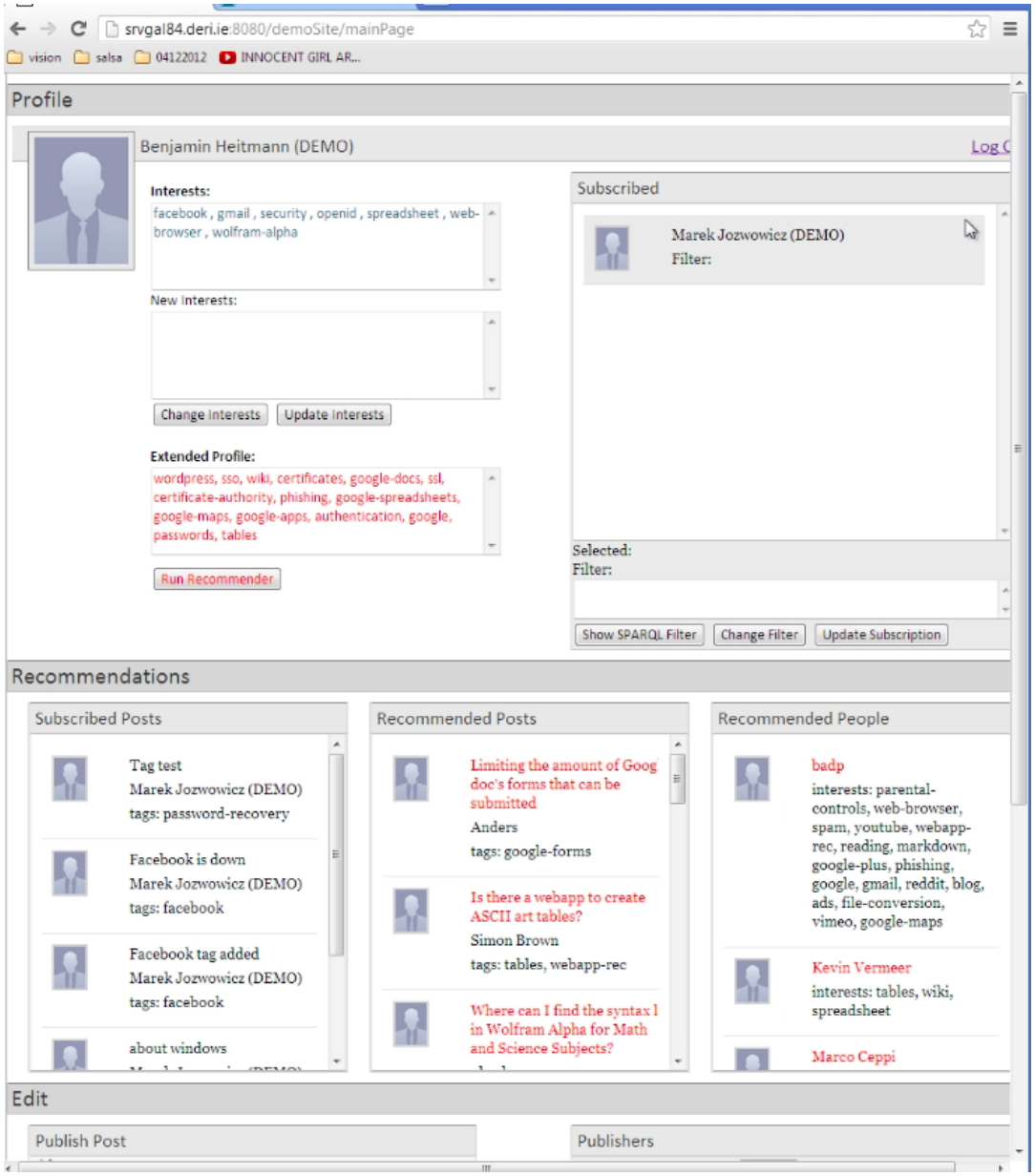

Figure 6.5.: User interface of the example connected social platform for the ADVANSSE prototype

Application logic: The application logic of the prototype has a very limited scope, as it mostly checks if newly created posts reuse existing tags.

XMPP client: The Java servlet which contains the web application with the user facing interface is connected to the ADVANSSE server via the Ignite Smack XMPP client library[25].

[25] http://www.igniterealtime.org/projects/smack/

RDF store: For the demonstrator we use Jena Fuseki[26] as a persistent store for the user generated content and user interests. Jena Fuseki was chosen as it provides a fast storage backend (Jena TDB), and because it provides a SPARQL HTTP end-point. The end-point is used by the Java servlet for querying and updating the stored RDF data.

6.5. Summary

In this chapter we have presented the architecture for the ADVANSSE distributed social platform and the ADVANSSE prototype that implements this architecture. Both the architecture and the prototype are the outcomes of the ADVANSSE collaboration project with CISCO Galway, which has the goal of providing a distributed platform for enabling personalised information access in an enterprise environment.

We first introduced the background regarding distributed organisational environments. The organizational landscape of enterprise IT environments shifts from a centralized environment with a small number of centralized content repositories to a more distributed model in which peers can both publish and consume information. The pre-requisite for providing personalisation for such distributed and domain-neutral user profiles from multiple sources, is to provide an infrastructure which enables aggregation and integration of user profile data from distributed social platforms. Towards this goal we discussed the differences between the pull, push and publish/subscribe interaction models. For the proposed ADVANSSE architecture, we choose the publish/subscribe model as it supports the greatest degree of decoupling of the three approaches.

We then introduced the three ADVANSSE use cases for knowledge sharing and discovery in a distributed enterprise environment. The use cases describe the following requirements: (1) filtering of messages using subscriptions; (2) providing recommendations for discovering content as it is published, without previous knowledge about people and their expertise; (3) allowing users to add new interests and receive previously published content related to their new interests.

We then propose the architecture for the ADVANSSE distributed social platform, which addresses these requirements. The proposed architecture can accommodate multiple ADVANSSE servers and multiple ADVANSSE connected social platforms.

Data synchronisation between all participating systems is enabled by the Extensible Messaging and Presence Protocol (XMPP) with the Publish/Subscribe extension, and the SPARQL Update protocol. XMPP allows messages to be transported and routed in a decentralised way. SPARQL update allows describing and transporting of graph data in the form of RDF, as part of the XMPP messages. By using XMPP and

[26] https://jena.apache.org/documentation/serving_data/index.html

SPARQL update together, the participating systems can exchange meta-data about published content, as well as user profile data. Each social platform is connected to one ADVANSSE server via XMPP, and if there is more than one ADVANSSE server, then ADVANSSE servers can be connected to each other.

ADVANSSE servers provide the functionality for message routing, aggregation of user profile data and content meta-data, and for personalisation using the aggregated user preferences.

ADVANSSE connected social platforms implement all of the user-facing functionality of the various social platforms.

The proposed architecture allows scaling the number of connected social platforms and servers as it uses XMPP to enable data synchronisation.

After proposing the architecture of the ADVANSSE distributed social platform, we describe the implementation of our prototype which implements both a server and a connected social platform according to the proposed architecture. For the prototype we use data which is as similar as possible to the data which is described in the use cases. Data about user profiles, tags and posts is provided by StackExchange, semantic background knowledge is provided by DBpedia, and the Cisco ERT vocabulary is used as the schema for representing and integrating all data into RDF. We implement the prototype ADVANSSE server using the Openfire XMPP server and Jena Fuseki project. We implemented our own baseline data homogenisation service. The personalisation component employs an implementation of SemStim using Java and the HDT RDF store. Our prototype implementation of an ADVANSSE connected social platform uses Java servlets and Apache Tomcat, as well as Jena Fuseki and the Ignite Smack XMPP library.

6.5.1. Contributions of this chapter

In this chapter we have addressed research questions (Q2) and (Q3):

Research question (Q2): How can an open ecosystem for personalisation be enabled as an alternative to closed ecosystems?

Research question (Q3): How can domain-neutral user profiles be implemented?

We have shown how the proposed architecture of the ADVANSSE distributed social platform supports an open ecosystem for personalisation (Q2), as it provides the infrastructure required for cross-domain personalisation using preference data from multiple sources and different domains. We also have shown how we have implemented domain-neutral user profiles (Q3) in the data homogenisation service of the ADVANSSE server in our prototype implementation. In addition, as the prototype uses the SemStim algorithms to make recommendations, we have described how to deploy the SemStim algorithm

for a real-world use case. Finally, the proposed architecture is an instantiation of the conceptual architecture for personalised Linked Data applications, which we presented in Chapter 3. By instantiating our conceptual architecture, we have shown by example how it applies to a real-world application.

Chapter 7.

Conclusion

In this thesis, we have presented an open framework for cross-domain personalisation. Our framework consists of two parts: (1) a conceptual architecture for recommender systems using Linked Open Data, and (2) a cross-domain recommendation algorithm, which is able to generate recommendations in a target domain using preferences in a different source domain.

Our framework addresses new requirements for personalisation approaches which arise from two recent architectural changes to recommender systems. These are the shift of recommender systems from closed to open inventories, and the emergence of ecosystems that provide the infrastructure for sharing of user profile data between personalised services.

We have shown that our framework is suitable for providing cross-domain recommendations, by evaluating the proposed cross-domain recommendation approach, and by instantiating a prototype based on our conceptual architecture. We demonstrated the ability of SemStim to provide cross-domain recommendations in an experiment using multi-rating data from Amazon.com. In the experiment, SemStim uses DVD preferences to recommend music and vice versa. In addition, we presented a prototype that is based on our conceptual architecture for Linked Open Data recommender systems. This prototype demonstrates how to deploy our cross-domain personalisation approach for a real-world use case and for real-world data.

We believe that ecosystems in which user profile data can be shared between personalised services will become more prevalent in the near future. Already today, Facebook has the role of a central hub in an ecosystem which allows external 3rd-party services to share user profiles with Facebook. If a user provides his permission, then one external service can access the user preferences aggregated by a second external service. Facebook is constantly expanding the amount of data that it collects from external sources, as can be seen by its most recent move in June 2014, to sell ads based on user preferences collected both on the Facebook site itself and on the partner sites, such as news papers

or search engines[1]. As Facebook is an internationally recognised leader in the field of social networking sites, other social networking operators such as Google and Twitter are sure to follow.

This implies that the number and the importance of such personalised ecosystems will only grow, which will increase the importance of generating cross-domain recommendations without the use of existing proprietary approaches. Our framework accomplishes this, as it describes how to provide cross-domain personalisation capabilities without the use of a proprietary and closed infrastructure, by using Linked Open Data and other open standards.

7.1. Summary of the thesis

In this thesis, we have first presented a conceptual architecture for Linked Open Data recommender systems, which describes how Linked Data can be used to share, aggregate and integrate multi-source and domain-neutral user preferences. We described our personalisation approach for cross-domain personalisation, which is based on SemStim, an unsupervised, graph-based algorithm for cross-domain personalisation. We described the evaluation of SemStim for both the single-domain and cross-domain recommendation task. We showed how our framework for cross-domain recommendations applies to a real-world use case from the industry. We described how we implemented a prototype as the outcome of the ADVANSSE[2] collaboration project with CISCO Galway. The ADVANSSE prototype is based on our conceptual architecture for Linked Open Data recommender systems, and showed how we deployed SemStim for real-world data.

We now describe each of our contributions in more detail. We discuss the limitations of our results in more detail in the next section.

7.1.1. Conceptual architecture for Linked Open Data recommender systems

In Chapter 3, we derived a conceptual architecture for Linked Open Data recommender systems.

In order to provide a strong empirical grounding for the conceptual architecture, we first performed an empirical survey of 124 applications. The applications come from the years 2003 to 2009 of the "Semantic Web challenge" and the years 2006 to 2009 of the "Scripting for the Semantic Web challenge".

[1] http://arstechnica.com/business/2014/06/facebook-brings-in-a-new-eerily-accurate-form-of-ad/
[2] http://advansse.deri.ie/

The findings of our survey show that support for the graph-based data model of RDF is virtually at 100% amongst the surveyed applications, and most applications reuse formal semantics such as the FOAF, DublinCore and SIOC vocabularies. Support for the Linked Data principles has reached more then half of the contest entries in 2009. Support for decentralised sources, data from multiple owners, and sources with heterogeneous formats is supported each year by a large majority of the surveyed applications. In addition, the majority of applications leverage Semantic Web technologies to support data integration.

We then used the results of the empirical survey as the empirical foundation for a component-based, conceptual architecture for Linked Data applications. The proposed conceptual architecture provides a template and best practices for the typical high level components required for providing personalisation in open ecosystems.

The components of the proposed conceptual architecture implement the functionality which is required to leverage Linked Open Data for personalisation in open ecosystems:

Multi-source user profiles are enabled by using the data discovery service component to aggregate distributed user profiles, and by using the data homogenisation service to integrate the different user profiles after they are aggregated. The resulting multi-source user profiles are then stored in the RDF store via the graph access layer or the graph query language service.

Domain-neutral user profiles are enabled by using the graph-based data model of RDF for representing the user profile, and by using concept identifiers from Linked Open Data to represent the user preferences from the different domains. The resulting domain-neutral user profile can then be stored in the RDF store via the graph access layer or the graph query language service.

Cross-domain personalisation is enabled by using an approach which can leverage the graph-based data model of RDF. The personalisation component then executes the recommendation algorithm, and uses the graph access layer to access background knowledge and the domain-neutral user profiles from the RDF store. The resulting recommendations are then passed on to the user interface or the graph-based navigation interface.

7.1.2. Cross-domain personalisation algorithm: SemStim

In Chapter 4, we described our personalisation approach for cross-domain personalisation, which is based on SemStim, our unsupervised, graph-based algorithm for cross-domain personalisation. It leverages multi-source, domain-neutral user profiles and the semantic network of DBpedia in order to generate recommendations for different source and target domains. SemStim does not require any overlap between the target and source domains,

and it does not require any ratings for the target domain. It is not affected by the cold-start problem, or by the availability or quality of meta-data about recommendable items, beyond the quality of linking items to DBpedia.

We then described the intuition behind the algorithm, and provided the formal details of the algorithm. The computational complexity of the SemStim algorithm is $O((|E| \log(|V|))^{\rho_{max}})$, where $|E|$ is the number of edges in the graph, $|V|$ is the number of vertices and ρ_{max} is the maximum number of phases. Each phase is a restart of the algorithm.

The high complexity of the algorithm can be explained by the ability to restart the algorithm and by the non-linear activation model. Restarting the algorithm if the required number of target activations has not been reached, allows exploring a larger subset of the graph, which however also increases the complexity. The non-linear activation model is induced by our choice of input, activation and output function, and allows SemStim to provide personalised results.

In practical terms, the time required to use SemStim for one user is around 2 orders of magnitude longer then using a CF algorithm. However, if no preferences are available for a user, then importing external user preference data in order to use SemStim for personalisation can be preferable to providing no personalisation for a user.

7.1.3. Evaluation of SemStim algorithm

In Chapter 5, we presented the results of evaluating our SemStim algorithm. Our evaluation consists of three different experiments. In addition, we describe the results of our participation in a recommender systems "implementation challenge".

We evaluated SemStim by comparing it to two other graph-based algorithms, Linked Data Semantic Distance (LDSD) and set-based breadth-first search (SetBFS). In addition, we compare SemStim to k-nearest neighbor collaborative filtering (CFknn), and SVD++, which is a collaborative filtering algorithm based on matrix factorisation. Lastly we use random selection as worst case baseline.

For our evaluation, we first use the MovieLens 100k data to compare the single-domain accuracy of the algorithms. Then we demonstrate the capability of SemStim to generate cross-domain recommendations, with different source and target domains. We do this by using profiles with ratings in two different domains provided by Amazon as part of a cross-domain rating data set. In addition, we evaluate the diversity of the recommendations generated by the different algorithms in two different experiments.

The results of the single-domain accuracy experiment show that the performance of SemStim dominates the performance of the other algorithms for the top-k recommendation task with equal source and target domains.

The results of the cross-domain accuracy experiment show that SemStim is able to generate cross-domain recommendations. When using DVDs as the source domain and music as the target domain, SemStim outperforms the other algorithms with statistically significant results. However, when reversing the source and target domain, so that music is used as the source domain and DVDs as the target domain, SemStim performs only as good as SetBFS, with a not statistically significant difference between the performance of the two algorithms.

The different performance of SemStim for these two different pairings of the domains music and DVDs indicates that the performance of SemStim is dependent on the selection of the specific target and source domain. As future work, we plan to investigate the connection between the performance of SemStim and e.g. the size of a domain or the connectedness of the items in a domain.

An additional external factor for the performance of SemStim in this experiment, is the quality of the linkage data between the items in the Amazon data set and DBpedia. As we only used basic heuristics to create this linkage data, the performance of SemStim might be improved by using a more sophisticated entity resolution approach. While the topic of entity consolidation and linking of entities to DBpedia is beyond the scope of this thesis, it provides a promising direction for future research.

Further, we performed an experiment to compare the diversity of the SemStim recommendations with those of the other algorithms. For this experiment we again used MovieLens 100k rating data as both source and target domain. To determine the diversity of the recommendations, we use our own feature-based diversity metric, which is based on estimating the number of clusters in a set of recommendations. The results of this experiment show that the diversity of recommendations generated by SemStim can be tuned by changing the activation threshold of nodes, which is one of the parameters of the algorithm. However, tuning the recommendations this way, requires using all of the preferences of a user, including his negative and neutral user preferences. When only using the positive preferences of a user, there is no correlation between the range of diversity and the activation threshold.

Then we described the results of our participation in the "diversity recommendation task" at the Linked Open Data-enabled recommender systems challenge at the Extended Semantic Web Conference 2014. We were able to show that SemStim has a competitive performance (3rd place out of 12) for the single-domain diversity recommendation task. The challenge organisers used the average of the ranks achieved for F1-score@20 and intra-list diversity (ILD@20) to rate the participants. In addition, we were able to show that for recommendations on the data set of the challenge, the diversity of the results of SemStim is inversely correlated to the accuracy of the results.

In summary, our evaluation showed that SemStim is able to provide cross-domain recommendations, which do not require any overlap between target and source domains, and without using any ratings in the target domain.

7.1.4. ADVANSSE use case and prototype implementation

In Chapter 6, we describe a prototype implementation that is the outcome of the ADVANSSE collaboration project with CISCO Galway. The prototype is based on our conceptual architecture for Linked Open Data recommender systems, and it shows how to implement all parts of our framework for cross-domain personalisation for a real-world use case and for real-world data.

We describe the implementation of our prototype that implements both a server and a connected social platform. For the prototype we use data that is as similar as possible to the data which is described in the ADVANSSE use cases. Data about user profiles, tags and posts is provided by StackExchange, semantic background knowledge is provided by DBpedia, and the Cisco ERT vocabulary is used as the schema for representing and integrating all data into RDF. We implement the prototype ADVANSSE server using the Openfire XMPP server and Jena Fuseki project. The personalisation component employs an implementation of SemStim using Java and the HDT RDF store. Our prototype implementation of an ADVANSSE connected social platform uses Java servlets and Apache Tomcat, as well as Jena Fuseki and the Ignite Smack XMPP library.

Our implementation demonstrates that the framework for cross-domain recommendation that we present in our thesis is well suited for practical application in current industry settings.

7.2. Directions for future research

We now present several directions for future research, which have emerged out of our research, but which are outside of the scope of this thesis.

Serendipitous discovery

The term "serendipity" means "fortunate happenstance"[3], and "serendipitous discovery" refers to discovery of interesting concepts, ideas or content by accident. SemStim implements associative retrieval of similar items in a semantic network, by using the direct and indirect connections between the starting nodes and the set of potential targets in a target domain. Associate retrieval is well suited for information retrieval tasks in

[3] https://en.wikipedia.org/wiki/Serendipity

which the exact kind of item which is required in order to satisfy the query can not be described in advance[4].

In this thesis, we have used SemStim only for the use case of personalisation in this thesis. However, the ability to perform retrieval by association can be used in other use cases, where serendipity is part of the requirements of the use case. Such use cases include personalisation of online news articles, or exploratory discovery of art in a virtual museum. We plan to investigate the ability of SemStim to be used for serendipitous discovery in our future research, by analysing user feedback in relation to serendipitous experiences.

Restrictions to the domains on which SemStim can be used

The results of the cross-domain accuracy experiment have shown, that pairs of domains exist for which the performance of SemStim is not symmetrical. In other words, SemStim performs better when using one of two domains as the target domain, and it will perform slightly worse, when the target and source domains become switched. In our experiment, SemStim had a better accuracy then the comparison algorithms when using DVDs as the source domain, and music as the target domain. When these target and source domains were switched, SemStim had the same accuracy as SetBFS, with the difference between the two algorithms being statistically insignificant.

This indicates that the performance of SemStim is dependent on the selection of the specific target and source domain. This raises the question: Which characteristics of the selected domains influence the performance of SemStim? Possible characteristics of a domain might be the connectedness of a domain, or the size of a domain. As future work, we plan to investigate the connection between the performance of SemStim and the choice of target and source domain.

Learning of weights for SemStim

One of the main limitations of the SemStim algorithm in its current form is that SemStim does not take the semantics of the different edge types into account. All algorithms derived from spreading activation can assign different weights to different edge types. However SemStim currently uses a uniform weight β for all edges in the graph. The main benefit of using a uniform weight, is that SemStim can be as an unsupervised approach for any combination of domains, without (a) any pre-processing of the graph and (b) without first requiring to learn weights depending on the choice of target and source domain.

[4] http://online.education.ed.ac.uk/gallery/gritton_serendipitous_learning/conclusion/assets/assignment_print_version.pdf

However, pre-processing of the graph using e.g. entropy measures as described by [Dehmer & Mowshowitz, 2011] are a promising direction for future research into improving the performance of the algorithm. The entry of a link type could be used to set the weight of that link type.

In addition, using machine learning approaches to learn the weight of different edge types, could be used in addition to pre-processing the graph. It might allow improving the performance of SemStim in general, and for specific domains. [Ostuni et al., 2013] can be seen as an example for integrating machine learning with a graph algorithm.

Quality of linkage data

The results of our cross-domain accuracy experiment suggest that the quality of linkage data between the items in the user profiles and DBpedia is an important factor for the performance of SemStim related to accuracy.

For this experiment, we used the Amazon multi-domain rating data set, which we had to link to DBpedia ourselves. We only used basic heuristics such as fuzzy full-text matching to create this linkage data. In addition, as many entities on DBpedia which correspond to books, music or movies are not correctly typed, many of the generated links point to concepts which have similar names, but which do not represent the correct type of item. For example, a childrens book called "My Very 1st Tea Party" was linked to the general concept of a "Tea party" (a gathering to drink tea).

On the one hand, this can be interpreted as an indicator for the robustness of SemStim to provide recommendations even if the linkage data contains errors. On the other hand, it seems likely that improving the quality of the linkage data, will have a positive impact on the recommendations of SemStim. Several mature approaches for linking entities to DBpedia exist, such as Kanopy [Hulpuş et al., 2013] and DBpedia Spotlight [Mendes et al., 2011].

While the topic of entity consolidation and linking of entities to DBpedia is beyond the scope of this thesis, it provides a promising direction for future research into improving the results of SemStim.

Appendix A.

An Architecture for Privacy-enabled User Profile Portability*

Providing relevant recommendations requires access to user profile data. Current social networking ecosystems allow third party services to request user authorisation for accessing profile data, as described in Chapter 2.

While the creation of such ecosystems provides powerful incentives for users to allow the sharing of their profile data between different services, it also leads to user lock-in and social networking data silos: User profiles are not portable between systems, connecting to users from a different system is not possible and the user can not evade changes to the terms of service.

In this chapter we propose an alternative: We present an architecture which describes how to combine existing infrastructure of the Web of Data and existing standards for decentralised identity management in order to achieve privacy-enabled user profile portability. Building on work by [Hollenbach et al., 2009], our architecture describes how to combine Linked Data, WebIDs and the Web Access Control (WAC) vocabulary:

- Linked Data [Bizer et al., 2009], and the Friend-of-a-Friend (FOAF) and Semantically Interlinked Online Communities (SIOC) vocabularies allow the description of domain independent user profiles [Bojars et al., 2008a].

*This chapter is based partially on previously published work:

1. Benjamin Heitmann, James G. Kim, Alexandre Passant, Conor Hayes, Hong-Gee Kim, "An architecture for privacy-enabled user profile portability on the Web of Data", In *Proceedings of the International Workshop on Information Heterogeneity and Fusion at the ACM Recommender Systems Conference*, 2010.

- WebIDs [Story et al., 2009] securely connect a user identity to the information in a user profile and can be used for authenticating a user.
- The WAC vocabulary [Hollenbach et al., 2009] allows the user to authorise third party services for accessing different parts of his profile information.

The proposed architecture enables the creation of a universal "private by default" ecosystem with interoperability of user profile data. The proposed architecture allows users to benefit from the privacy that is provided by centralised and closed social networking ecosystems as well as from the portability that is provided by the decentralised and open Web of Data. User profiles and activity stream data can be securely shared with any third party that supports the architecture. User profiles can be hosted by social networking sites or they can be self hosted by the user. There is no lock-in to any specific social networking site or ecosystem. We provide a qualitative evaluation of the presented architecture based on the evaluation framework for privacy-enhanced personalisation suggested by [Wang & Kobsa, 2009]. In addition we describe how the architecture applies to a use case from the e-Health domain.

The rest of the chapter is structured as follows: Section A.1 describes related work in identity management, distributed social networks and privacy-enhanced personalisation. Section A.2 lists the requirements for our architecture based on an existing evaluation framework for privacy-enhanced technologies. Section A.3 describes our architecture and the roles and communication pattern of it's participants. We also describe its application in an e-Health use case. Finally, Section A.4 provides a discussion of the proposed architecture.

A.1. Background: Personalisation and Privacy

Privacy and personalisation are currently at odds [Wang & Kobsa, 2009]. In order to provide a personalised experience it is necessary for a website to have access to data about that same user. However users can feel reluctant in sharing their personal data with a website, as they fear their data can be misused or traded with unknown entities. Different approaches to manage the identity and the profile data of the user have been suggested both inside of the recommender systems community as well as from the industrial Web standards community in general.

In addition to providing privacy features as part of a personalisation approach, the infrastructure used by the personalised service can be privacy-enabled. While the Web of

Data can provide sources of data and knowledge for personalised services, most Linked Data currently is public by default.

In this section we will first provide a short overview of privacy-enhanced personalisation approaches. Then we will introduce the most prominent current standards for identity management and decentralised social networking. Finally, we will discuss current sources of social data on the Web of Data, which are all public by default.

A.1.1. Privacy-enhanced personalisation

According to [Wang & Kobsa, 2009] the existing approaches for enhancing personalisation to enable user privacy fall into these categories:

Pseudonymous personalisation allows users to remain anonymous towards a personalised system, whilst enabling the system to still recognise the user in different sessions so that it can cater to the user personally. This also allows the user to keep different parts of his online activity on the same service apart (e.g. professional use and private use), as used by [Arlein et al., 2000].

Distributed personalisation, in which either the storage of the user data is distributed or the computation of the recommendation. This enables better privacy for the user, as each user controls the storage and the distribution of his own data. [Miller et al., 2004] propose a peer-to-peer algorithm called PocketLens. The algorithm first aggregates data from the direct neighbours in the P2P network and then generates an item-to-item similarity model based on this data. Then peers incrementally share the item-to-item model and use this to update their own model. This model then can be used to generate recommendations for a user.

Cryptography enhanced methods for personalisation treat the privacy-preserving computation of recommendations as secure multiparty computation problem, where users and different websites jointly conduct computations based on their private data without the need to trust each other. In order to achieve this the user data can be transmitted in an already encrypted state [Canny, 2002], it can contain randomised errors which cancelled out during the computation, or it can use obfuscation or aggregation to hide a single users preferences amongst the data of a group of users [Berkovsky et al., 2005].

Of these privacy-enhanced personalisation approaches, the work of [Arlein et al., 2000] from 2000 is most similar to our approach. Arlein presents an architecture for so-called

"global customisation", which enables third party services (called "merchants") to collect data about a user and share it between sites using a "profile database". Users are represented in the architecture as personas which are stored on a "persona server". Every user can have multiple personas, and the privacy of the user is protected because the profile database only knows about personas without being able to link personas to real users.

While our architecture has similar goals in allowing the sharing of user profile data for recommendations while maintaining the privacy of the user at the same time, we use different technologies for this which were not available in 2000. By combining existing standards and infrastructure to achieve these goals our architecture can be easily integrated into the emerging Web of Data from the start.

A.1.2. Decentralised social networking standards

Outside of research on privacy enhanced personalisation, standards have been developed for identity management and decentralised social networking. Some of these standards such as OpenID are widely used by now, while other standards are missing industry adoption, depending on the maturity of the standard. We introduce the most prominent industry standards for authentication, authorisation and profile data exchange in the following.

OpenID

OpenID[1] is a standard for decentralised authentication of a user. It provides a way to prove that an end user owns an identity URL without passing around the password of the user to a third party service. OpenID is completely decentralised meaning that anyone can choose to be a third party service ("consumer" in OpenID terminology) or hub site ("identity provider") without having to register or be approved by any central authority. Users can pick which hub site they wish to use and preserve their identity as they move between hub sites. As of December 2009, there are over 1 billion OpenID enabled accounts and approximately 9 million sites have integrated OpenID consumer support, such as Google and Yahoo.

OpenID provides the means to decouple identities from real users, thus enabling pseudonymous personalisation. However OpenID is not well suited for machine agents

[1] http://openid.net/specs/

and it requires a large overhead in terms of the number of HTTP connections which are required to gain access to a secured resource [Story et al., 2009].

OAuth

OAuth[2] specifies a protocol for decentralised authorisation of resource access. It specifies how a user can authorise a third party service (called "client" in OAuth terminology) to access parts of his profile (his "resources") on a hub site ("resource owner"). Instead of using a user's password to access parts of his profile at a hub site, third party services obtain access tokens which are used to access the protected resources. In addition to removing the need for users to share their password, users can restrict access to a limited part of their data and they can limit access duration.

OAuth is used by two of the most popular current social websites (Twitter and Facebook) to authorise third party services for accessing data from user profiles. OAuth complements OpenID, as it provides delegation of authorisation on top of the authentication through OpenID identity URLs. However this thight integration also means that OAuth requires a multitude of HTTP connections, which will lead to scalability problems for decentralised authorisation on the Web of Data.

OpenID attribute exchange

OpenID itself does not provide any mechanism to exchange profile information, although it is possible to link an OpenID identity to a profile data such as a vCARD document. However **OpenID attribute exchange**[3] provides a protocol accessing profile data and provides a data model for storing it. The OpenID attribute exchange standard has not reached industry wide adoption, as it specifies a very limited vocabulary for expressing a user profile and it does not allow easy extensions to this vocabulary.

A.1.3. The Web of Data: Public by Default

Social websites provide a big contribution to the linked data cloud, by making information about their users available. This data is modelled after the principle of object centred sociality [Bojars et al., 2008a]: persons are not only directly connected to other persons, but also indirectly via objects of a social focus. In this way a community is connected

[2] http://tools.ietf.org/html/draft-ietf-oauth-v2
[3] http://openid.net/srv/ax/1.0

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 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 [Bojars et al., 2008a].

FOAF and SIOC provide the means for putting user profiles and data about object centred sociality on the Web of Data. However this data is usually "public by default", as the Web of Data does not currently allow the user to specify which services can access which parts of a user profile.

However, as the recent Facebook privacy backlash [Fletcher, 2010] has shown, users are not comfortable with the assumption that all of their profile data is publicly accessible. Facebook launched in 2004 with a strong "private by default" policy, however it tried to move away from this in 2010 towards a "public by default" policy. This move caused a backlash with many high profile users cancelling their account. This led to Facebook reverting their policy and now suggesting more private settings again. See [Boyd & Hargittai, 2010] for more background.

In order to provide incentives for users to share their profile with different recommendation services on the Web of Data, it is necessary to provide the means for controlling the access to the profile data. This will allow users to move towards a "private by default" policy, ultimately leading to more user profiles being available to recommendation services.

A.2. Requirements analysis

In order to arrive at an architecture for privacy-enabled user profile portability for the purpose of making recommendations, we collected requirements for the architecture. The requirements are based on the evaluation framework for privacy enhancing technologies (PETs) by [Wang & Kobsa, 2009] [Kobsa, 2007]. We first outline their list of general privacy principles and the main areas in which users have privacy concerns. Then we describe our non-functional requirements for the architecture which are informed by the emergence of the Web of Data.

A.2.1. Privacy principles

As part of their evaluation framework, [Wang & Kobsa, 2009] identify the main privacy principles which motivate the creation of privacy enhancing technologies. These privacy principles are grouped as follows:

anonymity related principles from the security literature. These include anonymity, pseudonymity, unobservability, unlinkability and deniability.

privacy principles from privacy laws, regulations and recommendations. These principles have been identified from a review of 40 international privacy laws, and they include limiting the collection of data, specifying the explicit purpose of the collection, limiting the use of the data for specific functions of the service, and informing the user of onward transfer of the data to third part services, as well as asking the user for his consent.

human-computer interaction for the purpose of enabling privacy. These include asking the user for his privacy preferences, allowing the user to negotiate by giving him multiple privacy choices and the usability of the service by e.g. not requiring installation of new infrastructure software from the user.

A.2.2. Privacy concerns

[Kobsa, 2007] also identify the main privacy concerns which users have regarding the impact of technology on the previously identified privacy principles. The concerns of users about their privacy fall roughly into three areas:

the protection of identity Users want to control who can identify them, or who can link their identities on the Web back to their official and legal identity. This corresponds to enabling and protecting the anonymity related principles.

control over the user's data In addition to controlling their identity, users value the ability to control who can access which parts of their profile data. For our purposes this not only includes mostly static information like name, gender and location, but also the highly dynamic activity stream of the user and all the multimedia resources associated with the user. This area is affected by enabling and protecting the privacy principles collected from laws, regulations and recommendations.

human-computer interaction Different aspects of enabling the protection of identity and the control of a user over his data depend on the user interface of a service. For

instance, by law a user needs to be informed of the data which is collected of him, which is a task of the UI. In addition the UI has the task to community all of the possible privacy settings without e.g. hiding some of them in a complicated menu structure. This area is affected by enabling and protecting the privacy principles collected from laws and those principles collected from human-computer interaction research.

A.2.3. Non-functional requirements

In addition to the requirements imposed by the privacy concerns, we have identified non-functional requirements for our architecture. While these do not directly impact the privacy of the user, it is necessary to take them into account to make large scale adoption of the architecture possible.

universality The World Wide Web was perceived to be one universal space[4], where any resource can be connected to any other resource. However for user identities on the current generation of social websites this is not true, as one user can only be connected to other users from the same social network. Therefore our architecture should enable universality of user profiles.

scalability An architecture for portable user profiles should scale well for the number of users in total, as well as for the number of hub sites and third party services. Users might have one or multiple profiles. Hub sites might be created just for one user if the user decides to host his user profile himself, or they might contain data from millions of users. Third party services might access data from any number of hub sites or individually hosted user profiles.

reuse of infrastructure Deploying an architecture for portable user profiles should not depend on new backend infrastructure or on new client software on the side of the users. Ideally existing infrastructure from the World Wide Web such as Web servers using HTTP and URIs should be reused. In addition technologies from the emerging Web of Data can be extended for user profile portability.

[4] http://www.w3.org/DesignIssues/Axioms.html

A.3. Proposed architecture

In order to enable users to share their profiles with different ecosystems while maintaining their privacy at the same time, it is necessary to define an architecture for privacy-enhanced user profile portability. This architecture prescribes the standards as well as the roles and the communication pattern between the different participants. By implementing this architecture, all *individual* participants agree on the same technical principles, which in turn allows the architecture to guarantee the identified requirements on a *global* level.

In this section we first describe the Semantic Web standards and technologies which provide the foundation for our architecture. Then we describe the roles performed by participants of the architecture and introduce a use case grounding it in the e-Health domain. Based on the use case we describe the required communication pattern of the participants, followed by a qualitative evaluation against the identified requirements from the previous section.

A.3.1. Foundation standards

[Hollenbach et al., 2009] suggest using FOAF, WebIDs and the Web Access Control (WAC) vocabulary to enable access control in collaborative environments on the Web of Data. This allows integrating with existing infrastructure thereby extending the Web of Data in a natural way.

FOAF Vocabulary The FOAF vocabulary allows the description of domain independent user profiles [Bojars et al., 2008a]. FOAF provides properties to describe all of the details which are usually contained in a social networking profile or on a personal homepage. In addition a FOAF profile provides a container for other information from different domains. For instance, this information could use the SIOC vocabulary to list the content which the user has generated on his blog, on his twitter stream and the comments on different forums.

WebIDs [Story et al., 2009] securely connect a user identity to the information in a user profile and can be used for authenticating a user. A WebID consists of two parts: (1) an SSL certificate which contains a link to (2) the URI from which information about the user can be obtained. The data which is obtained from the URI is associated in return with the SSL certificate, as it lists the public key which is associated with the private key contained in the SSL certificate.

Web Access Control (WAC) vocabulary [Hollenbach et al., 2009] allows the user to authorise third party services for accessing different parts of his profile information. Each private resource is tied to an Access Control List (ACL) resource. The ACL resource can say which agents or groups of agents have access rights to the resources it governs, so the content of the ACL resource can be considered as a whitelist (i.e., "*private by default*").

Users are granted full access to the ACL resources for their profile so users can *read*, *write*, and *control* their whitelists as well as the profiles themselves. In other words, users can update their profiles and they can also give third party services access to all or parts of their profile data.

A.3.2. Roles

The interplay between FOAF, WebIDs and the WAC vocabulary requires the participants to perform one of three roles: profile storage services, data consumers and user agents.

Profile storage services roughly correspond to the hub sites in current profile sharing ecosystems. They provide the storage for the user profile or parts of it, and they secure the access to the profile data by following the rules from the ACL about a profile. In addition they provide a user interface for changing and maintaining the ACLs from the WAC metadata by e.g. adding or removing read rights for data consumers. Profile storage services can be either self hosted by the user or they can be hosted by a social networking site.

Data consumers correspond to any type of third party service which is accessing user profile data in current ecosystems. Each consumer has its own WebID, which identifies the service every time it is accessing profile data from a profile storage service. This allows the storage to determine if the access is granted to the consumer.

User agents manage the different identities of a user. Each identity is represented by a WebID, which is used for authenticating the user towards profile storage service or data consuming services.

A.3.3. Use case: personal health records

In order to illustrate how the standards of FOAF, WebIDs and WAC vocabulary enable privacy preserving profile portability, we will ground the architecture in a use case from the e-Health domain.

The emergence of Personally Controlled Health Record (PCHR) platforms such as Google Health[5], Microsoft HealthVault[6], and Dossia[7] leads to "tectonic shifts in the health information economy" [Mandl & Kohane, 2008]. PCHR platforms enable patients to import and manage their health data, and third party services to securely access it and to provide added value. For instance, patients can get recommendations of clinical trials matching their condition, or they can exchange experiences with other patients having the same disease. However, while being central hub sites, these platforms also contribute to the data silo problem and the privacy of health data is required to be protected by law.

In order to meet these requirements, our proposed architecture can be applied to this domain, creating a Personal Health Application (PHA) ecosystem around portable health data without compromising privacy. As a result, third party services like TrialX[8] and PatientsLikeMe[9] can utilise patient (user) profiles with their consent, including their health data, to provide personalised information recommendations.

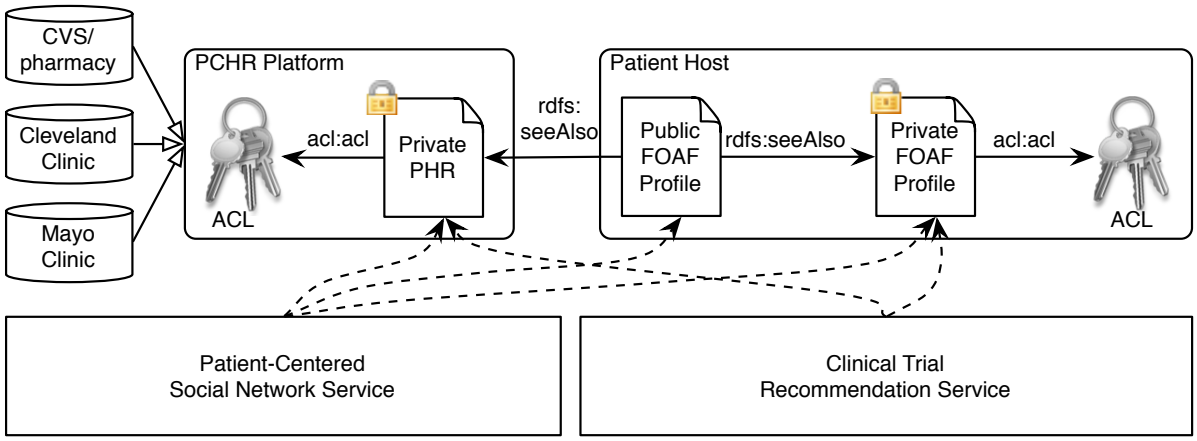

Figure A.1.: Architecture of the use case

In this use case, the patient profile is divided into three different resources, as shown in Figure A.1:

[5] http://health.google.com
[6] http://www.healthvault.com
[7] http://www.dossia.org
[8] http://trialx.com/
[9] http://www.patientslikeme.com/

Public FOAF Profile which is also used for WebID authentication and self hosted by the patient. This resource contains the public key of the certificate and pointers to the other resources, as well as the same kinds of public information social networking sites often provide, such as name, gender, and date of birth.

Private Personal Health Record (PHR) hosted by a PCHR platform. This contains the same kinds of information which would traditionally be found in a patients' medical record. This includes a patients' health conditions, medications, and laboratory results. The PHR is described by using the Health Level 7 Clinical Document Architecture (CDA), which is a widely used standard for exchanging patient health data. CDA specifies the semantics of PHR as well as the structure, so it can be converted into RDF triples and semantically queried [Liu et al., 2009].

Private FOAF profile which contains a list of friends who share the same or a similar disease with the patient. This patient-centered social network can be used for exchanging experiences about the treatment or the symptoms between patients having the same or similar diseases. This resource is also self hosted by the patient.

The different profiles can be implemented either as separate URIs and documents, which are discoverable via the main public profile, or they can be accessible via the same WebID, depending on the credentials of the WebID associated with the requesting user [Abel et al., 2007].

As the PHR is a rich source of information about a patient, it will be a valuable asset for third party services to provide personalised information recommendations. For example, a service similar to TrialX can utilise health data in the profile to give the patient recommendations on which clinical trials are matched to him/her. To this end, the patient has to grant the service access to the PHR in his/her profile by adding a new entry to the ACL.

In the same way, a service like PatientsLikeMe can gain exclusive access to the private FOAF profile which contains the social network of patients who suffer from the same or a similar disease. This kind of service can then give recommendations on care options based on the treatment of similar patients without compromising the privacy of the patient (e.g., information about the disease will not be accessible to the work related social network of the patient).

A.3.4. Communication pattern

In order for user agents, data consuming services and profile storage services to participate in the architecture, they need to interact with each other according to their role. In the following we describe an example of the resulting communication pattern as applied to the use case in section A.3.3. The communication pattern is illustrated in figure A.2.

1. The patient goes to the third party service using an HTTPS connection. During the SSL handshake, the patient sends his/her WebID to the service. The service verifies the identity of the patient by comparing the public key in the certificate with the one in the public FOAF profile.
2. The third party service welcomes the patient, and asks for permission to access the patient's PHR.
3. When the patient say "yes" to the request, he/she is getting sent to the PCHR platform wherein his/her PHR resides, then the PCHR platform authenticates the patient via WebID and authorises him/her to add a new entry to the ACL for his profile.
4. The PCHR platform asks the patient to confirm the new ACL entry, which specifies that the third party service denoted by the user specified WebID can now access to the patient's PHR.
5. Then the patient confirms the creation of the ACL entry.
6. When the third party service tries to access to the patient's PHR data, the PCHR platform verifies the WebID of the consuming service and checks the ACL of the patient's PHR data. If the permission of the consuming service to access the patient's PHR data is verified, access is granted and the consuming service can perform read operations on the data.
7. After the service gets the patient's PHR data, it compares the patient's health data with clinical trials and recommends the matched ones to the patient.

A.3.5. Qualitative evaluation

In order to evaluate the presented architecture, we now describe how it meets the identified requirements from section A.2:

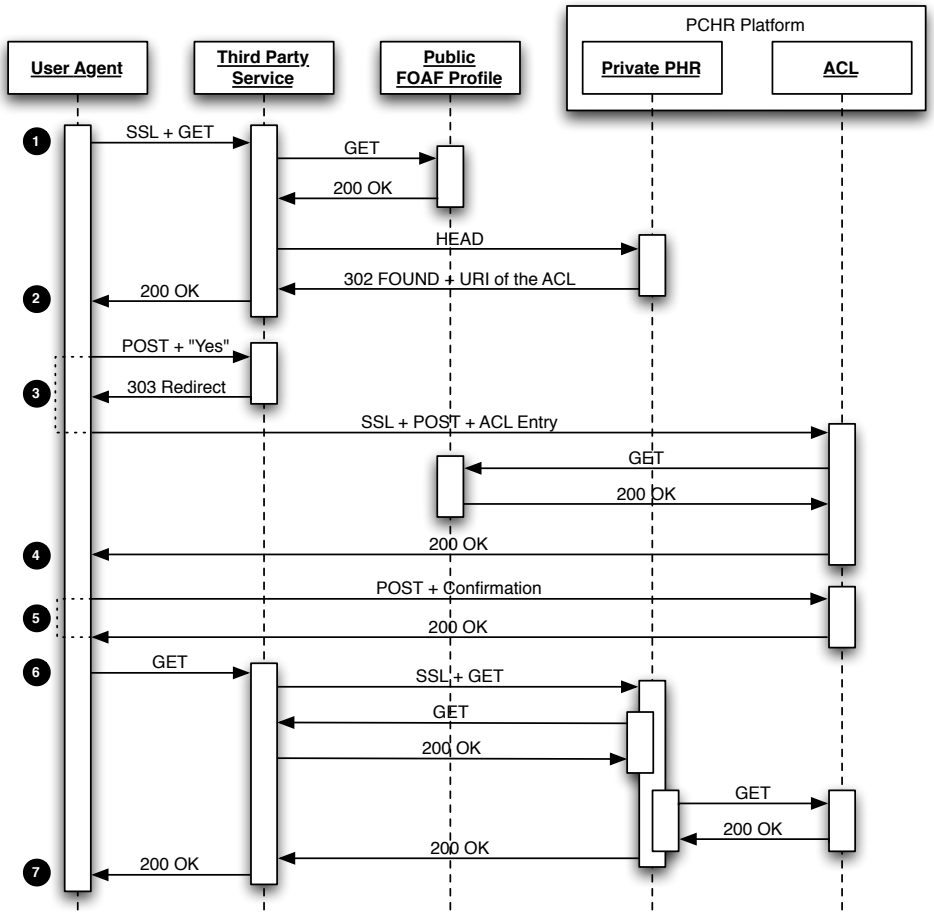

Figure A.2.: Sequence diagram depicting the communication pattern between the user agent, the PCHR platform and the third party service

Protection of identity Users can choose to use multiple identities, each identity being represented by a unique WebID. Each time a user interacts with a data consuming service his user agent can allow him to choose which WebID to use. In this way pseudonymity, unobservability and deniability of the user identity are supported. None of the identities need to be tied to a real world identity, thus supporting anonymity. Data consumers should not be able to link user identities, however user profile storage services need to be trusted in order to maintain unlinkability of user identities. Self hosting of a user's profiles however can impact the protection of his identities negatively, if the server can be easily linked to a real world identity.

Control over the user data The user stays in control of his profile data, as the portability of user profiles allows him to move his profile freely between storage services or even to host the storage of his profiles on his own server. Lock-in to a specific ecosystem or to a specific storage service should not be possible, as the open standards of RDF, FOAF and SIOC are used for describing the profile.

Human-computer interaction Services can provide an easy user interface for managing the ACL of a user profile. A user interface example is given in [Hollenbach et al., 2009]. The proposed architecture does not require the user to install or understand new software. WebIDs can be used in contemporary Web browsers such as Firefox, which support the installation of user generated SSL certificates.

Non-functional requirements The presented architecture allows any user agent, profile storage service or data consumer to participate in one universal ecosystem, as all participants will support the same standards and implement the same communication pattern. The architecture is scalable, as there are no bottlenecks or central points of failure, due to the decentralised nature of the used standards. For profile storage and data consumption existing standards and infrastructure from the World Wide Web and the Web of Data, such as HTTP and RDF are reused, thus making future adoption by service providers easy.

This shows that the proposed architecture allows us to create an ecosystem in which users can protect their identity, they have control over their own data and the interaction with the technology is easily understandable. The architecture also has the properties of universality, scalability and reuse of existing infrastructure. This allows users to benefit from an ecosystem which provides privacy and security while enabling portability of user identities at the same time.

A.4. Discussion

To be able to provide accurate recommendations by seamlessly combining multiple sources of information is a key objective of recommender system research. However, it invariably raises questions about data ownership and control. By seeking better data fusion techniques, are we tacitly acknowledging that we will ride roughshod over user privacy in order to build richer user models? Furthermore, we need to ask who will own and control the user models and how we may prevent them from being abused. The recent resistance to Facebook's change in its terms of service suggest that Web users are sensitive to how their data is used.

We argue that innovations in reconciling heterogeneous data sources must also be matched by innovations in architecture design and recommender methodology. The problem is simply not one of creating a richer, centralised database on which to create innovative new models, but of designing flexible recommender systems that can cater to differing degrees of data access and control.

Previous research has established the importance of maintaining user trust [Chellappa & Sin, 2005b] for successful adoption of personalisation technology. To build systems in which engendering user trust is a fundamental principle requires researchers to solve the challenge of data integration for recommendation systems by algorithmic, architectural and policy-based innovations.

Our approach does not address data integration by examining new data fusion techniques. Instead we point to the Web of Data as an example of how heterogeneous data sources can be linked by current standards. However, the Web of Data still requires techniques to allow data providers control access to their data. As such, our focus is a simple personalisation architecture, created from existing standards, that enables user profile portability and cross-domain recommendation while also allowing the user control of his data. The focus of current ecosystems is on locking data into a central hub site, and not on empowering the individual user. Our proposed architecture enables a universal ecosystem which is built around the user profile, by specifying an interoperable way to share and protect the profile data at the same time. Extending the infrastructure of current ecosystem is feasible, as our presented architecture naturally builds on existing standards of the World Wide Web and the Web of Data, such as HTTP and RDF.

A.5. Chapter summary

In this chapter we addressed the problem of preserving user privacy while seeking to integrate multiple personal information sources. The default architectural solution requires a centralised hub with users reliant upon the good will of the service provider to 'do no evil'. We argue that in order to maintain user trust and support the challenge of heterogeneous data integration must be addressed by algorithmic, architectural and policy-based innovations. As such, we presented an architecture for privacy-enabled user profile portability based on existing standards. We described the requirements for privacy-enabled sharing of profile data for data-mining and modeling. Finally, we illustrated our approach with a use case from the e-Health domain.

A.5.1. Contributions of this chapter

The contributions of this chapter are:

- An architecture for privacy-enabled user profile portability.

- A list of the requirements for privacy-protected sharing of profile data for the purpose of data mining and recommendations.
- A use case which shows how to apply the described architecture to the e-Health domain.

Bibliography

[cro, 2005] (2005). CROSI - Capturing, Representing and Operationalising Semantic Integration, The University of Southampton and Hewlett Packard Laboratories. http://eprints.ecs.soton.ac.uk/10842/1/crosi-survey.pdf.

[Abecker & van Elst, 2004] Abecker, A. & van Elst, L. (2004). *Handbook on Ontologies in Information Systems*, chapter Ontologies for knowledge management, (pp. 453–474). Springer.

[Abel et al., 2007] Abel, F., De Coi, J., Henze, N., Koesling, A., Krause, D., & Olmedilla, D. (2007). Enabling Advanced and Context-dependent Access Control in RDF Stores. In *International Semantic Web Conference*.

[Abel et al., 2011] Abel, F., Herder, E., Houben, G., Henze, N., & Krause, D. (2011). Cross-system user modeling and personalization on the social web. *User Modeling and User-Adapted Interaction (UMUAI)*, 22(3), 1–42.

[Adomavicius & Tuzhilin, 2005] Adomavicius, G. & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6), 734–749.

[Aimeur et al., 2006] Aimeur, E., Brassard, G., Fernandez, J. M., & Onana, F. (2006). Privacy-preserving demographic filtering. In *Proceedings of the 2006 ACM symposium on Applied computing* (pp. 872–878).: ACM.

[Amatriain & Basilico, 2012] Amatriain, X. & Basilico, J. (2012). Netflix Recommendations: Beyond the 5 stars (Part 2). The official Netflix Tech Blog.

[Amini et al., 2011] Amini, B., Ibrahim, R., & Othman, M. (2011). Discovering the impact of knowledge in recommender systems: A comparative study. *International Journal of Computer Science and Engineering Survey (IJCSES)*, 2(3), 1–14.

[Arenas & Pérez, 2011] Arenas, M. & Pérez, J. (2011). Querying semantic web data with sparql. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '11 (pp. 305–316). New York, NY, USA: ACM.

[Arlein et al., 2000] Arlein, R., Jai, B., Jakobsson, M., Monrose, F., & Reiter, M. (2000). Privacy-preserving Global Customization. In *Proceedings of the 2nd ACM conference on Electronic commerce* (pp. 176–184).: ACM.

[Auer et al., 2009] Auer, S., Lehmann, J., & Hellmann, S. (2009). Linkedgeodata: Adding a spatial dimension to the web of data. In *The Semantic Web-ISWC 2009* (pp. 731–746).

Springer.

[Balabanovic & Shoham, 1997] Balabanovic, M. & Shoham, Y. (1997). Fab: Content-Based, Collaborative Recommendation. *Communications of the ACM*, 40(3), 67.

[Barragáns-Martínez et al., 2010] Barragáns-Martínez, A. B., Costa-Montenegro, E., Burguillo, J. C., Rey-López, M., Mikic-Fonte, F. A., & Peleteiro, A. (2010). A hybrid content-based and item-based collaborative filtering approach to recommend tv programs enhanced with singular value decomposition. *Information Sciences*, 180(22), 4290–4311.

[Becker & Bizer, 2008] Becker, C. & Bizer, C. (2008). DBpedia Mobile: A Location-Aware Semantic Web Client. In *Semantic Web Challenge at the International Semantic Web Conference*.

[Bell & Koren, 2007] Bell, R. M. & Koren, Y. (2007). Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2), 75–79.

[Bennett & Lanning, 2007] Bennett, J. & Lanning, S. (2007). The netflix prize. In *Proceedings of KDD Cup and Workshop*, volume 2007 (pp.35).

[Berkovsky et al., 2005] Berkovsky, S., Eytani, Y., Kuflik, T., & Ricci, F. (2005). Privacy-enhanced Collaborative Filtering. In *Proc. User Modeling Workshop on Privacy-Enhanced Personalization*.

[Berners-Lee, 2006] Berners-Lee, T. (2006). Linked Data - Design Issues. http://www.w3.org/DesignIssues/LinkedData.html.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. *Scientific american*, 284(5), 28–37.

[Berthold et al., 2009] Berthold, M., Brandes, U., Kötter, T., Mader, M., Nagel, U., & Thiel, K. (2009). Pure spreading activation is pointless. In *Conf. on Information and knowledge management*.

[Bhide et al., 2002] Bhide, M., Deolasee, P., Katkar, A., Panchbudhe, A., Ramamritham, K., & Shenoy, P. (2002). Adaptive push-pull: Disseminating dynamic web data. *IEEE Transactions on Computers*, 51, 652–668.

[Bizer, 2009] Bizer, C. (2009). The Emerging Web of Linked Data. *IEEE Intelligent Systems*, (pp. 87–92).

[Bizer et al., 2007] Bizer, C., Cyganiak, R., & Heath, T. (2007). *How to Publish Linked Data on the Web*. Technical report, FU Berlin.

[Bizer et al., 2009] Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3), 1–22.

[Bizer & Schultz, 2009] Bizer, C. & Schultz, A. (2009). The berlin sparql benchmark. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(2), 1–24.

[Bobadilla et al., 2013] Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*.

[Bōjars et al., 2007] Bōjars, U., Heitmann, B., & Oren, E. (2007). A Prototype to Explore Content and Context on Social Community Sites. In *Proceedings of the International Conference on Social Semantic Web (CSSW)*.

[Bojars et al., 2008a] Bojars, U., Passant, A., Breslin, J., & Decker, S. (2008a). Social network and data portability using semantic web technologies. In *2nd Workshop on Social Aspects of the Web (SAW 2008) at BIS2008* (pp. 5–19).: Citeseer.

[Bojars et al., 2008b] Bojars, U., Passant, A., Cyganiak, R., & Breslin, J. (2008b). Weaving sioc into the web of linked data. In *Linked Data on the Web Workshop*.

[Boyd & Hargittai, 2010] Boyd, D. & Hargittai, E. (2010). Facebook Privacy Settings: Who cares? *First Monday*, 15(8).

[Breslin et al., 2006] Breslin, J., Decker, S., Harth, A., & Bojars, U. (2006). Sioc: An approach to connect web-based communities. *The International Journal of Web-Based Communities*.

[Broekstra et al., 2002] Broekstra, J., Kampman, A., & van Harmelen, F. (2002). Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *International Semantic Web Conference*.

[Brusilovsky & Henze, 2007] Brusilovsky, P. & Henze, N. (2007). *The Adaptive Web: Methods and Strategies of Web Personalization. Lecture Notes in Computer Science*, chapter Open corpus adaptive educational hypermedia. Springer.

[Burke, 2002] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4), 331–370.

[Candillier et al., 2007] Candillier, L., Meyer, F., & Boullé, M. (2007). Comparing state-of-the-art collaborative filtering systems. In *Machine Learning and Data Mining in Pattern Recognition* (pp. 548–562). Springer.

[Canny, 2002] Canny, J. (2002). Collaborative Filtering with Privacy. In *IEEE Symposium on Security and Privacy*.

[Cardoso, 2007] Cardoso, J. (2007). The Semantic Web Vision: Where Are We? *IEEE Intelligent Systems*, 22, 84–88.

[Chellappa & Sin, 2005a] Chellappa, R. & Sin, R. (2005a). Personalization versus Privacy: An Empirical Examination of the Online Consumers Dilemma. *Information Technology and Management*, 6(2), 181–202.

[Chellappa & Sin, 2005b] Chellappa, R. & Sin, R. (2005b). Personalization versus Privacy: An Empirical Examination of the Online Consumers Dilemma. *Information Technology and Management*, 6(2), 181–202.

[Cohen & Kjeldsen, 1987] Cohen, P. & Kjeldsen, R. (1987). Information retrieval by constrained spreading activation in semantic networks. *Information processing &*

management, 23(4), 255–268.

[Cremonesi et al., 2010] Cremonesi, P., Koren, Y., & Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In *ACM Conference on Recommender Systems* (pp. 39–46).

[Cremonesi et al., 2011] Cremonesi, P., Tripodi, A., & Turrin, R. (2011). Cross-domain recommender systems. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on* (pp. 496–503).: IEEE.

[Crestani, 1997] Crestani, F. (1997). Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, 11(6), 453–482.

[Cunha & de Lucena, 2006] Cunha, L. M. & de Lucena, C. J. P. (2006). *Cluster The Semantic Web Challenges Applications: Architecture and Metadata Overview*. Technical report, Pontificia Universidade Catolica do Rio de Janeiro.

[Cunha & de Lucena, 2007] Cunha, L. M. & de Lucena, C. J. P. (2007). *A Semantic Web Application Framework*. Technical report, Pontificia Universidade Catolica do Rio de Janeiro.

[Dadzie & Rowe, 2011] Dadzie, A.-S. & Rowe, M. (2011). Approaches to visualising Linked Data: A survey. *Semantic Web*, 2(2), 89–124.

[Davies et al., 2002] Davies, J., Fensel, D., & van Harmelen, F., Eds. (2002). *Towards the Semantic Web: Ontology-driven Knowledge Management*. John Wiley and Sons.

[Decker et al., 2000] Decker, S., Melnik, S., Van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., Erdmann, M., & Horrocks, I. (2000). The semantic web: The roles of XML and RDF. *IEEE Internet computing*, 4(5), 63–73.

[Dehmer & Mowshowitz, 2011] Dehmer, M. & Mowshowitz, A. (2011). A history of graph entropy measures. *Information Sciences*, 181(1), 57 – 78.

[Doan & Halevy, 2005] Doan, A. & Halevy, A. Y. (2005). Semantic integration research in the database community: A brief survey. *AI Magazine*, (pp. 83–94).

[Du & Brewer, 2008] Du, B. & Brewer, E. A. (2008). Dtwiki: a disconnection and intermittency tolerant wiki. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08 (pp. 945–952). New York, NY, USA: ACM.

[Eugster et al., 2000] Eugster, P., Guerraoui, R., & Sventek, J. (2000). "Distributed Asynchronous Collections: Abstractions for Publish/Subscribe Interaction". In E. Bertino (Ed.), *ECOOP 2000 Object-Oriented Programming*, volume 1850 of *Lecture Notes in Computer Science* (pp. 252–276). Springer Berlin, Heidelberg.

[Eugster et al., 2003] Eugster, P. T., Felber, P. A., Guerraoui, R., & Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM Comput. Surv.*, 35, 114–131.

[Fensel et al., 2001] Fensel, D., Van Harmelen, F., Horrocks, I., McGuinness, D., & Patel-Schneider, P. (2001). OIL: An ontology infrastructure for the semantic web. *IEEE intelligent systems*, 16(2), 38–45.

[Fernández et al., 2013] Fernández, J. D., Martínez-Prieto, M. A., Gutiérrez, C., Polleres, A., & Arias, M. (2013). Binary RDF Representation for Publication and Exchange (HDT). *Web Semantics*, 19(2).

[Fernández-Tobías et al., 2012] Fernández-Tobías, I., Cantador, I., Kaminskas, M., & Ricci, F. (2012). Cross-domain recommender systems: A survey of the state of the art. In *Spanish Conference on Information Retrieval*.

[Fielding, 2000] Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, UC Irvine.

[Fletcher, 2010] Fletcher, D. (2010). How Facebook Is Redefining Privacy. Time Magazine.

[Gantner et al., 2011] Gantner, Z., Rendle, S., Freudenthaler, C., & Schmidt-Thieme, L. (2011). Mymedialite: A free recommender system library. In *Proceedings of the fifth ACM conference on Recommender systems* (pp. 305–308).: ACM.

[Garcia-Castro et al., 2008] Garcia-Castro, R., Gómez-Pérez, A., Muñoz-García, Ó., & Nixon, L. J. B. (2008). Towards a Component-Based Framework for Developing Semantic Web Applications. In *Asian Semantic Web Conference*.

[Gold et al., 2001] Gold, A. H., Malhotra, A., & Segars, A. H. (2001). Knowledge management: An organizational capabilities perspective. *J. Manage. Inf. Syst.*, 18, 185–214.

[Griffin & Flanagan, 2010] Griffin, K. & Flanagan, C. (2010). Evaluation of asynchronous event mechanisms for browser-based real-time communication integration. In K. Elleithy, T. Sobh, M. Iskander, V. Kapila, M. A. Karim, & A. Mahmood (Eds.), *Technological Developments in Networking, Education and Automation* (pp. 461–466). Springer Netherlands.

[Halkidi et al., 2001] Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3), 107–145.

[Harth et al., 2010] Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.-U., & Umbrich, J. (2010). Data Summaries for On-demand Queries over Linked Data. In *World Wide Web Conference*.

[Heino et al., 2009] Heino, N., Dietzold, S., Martin, M., & Auer, S. (2009). Developing semantic web applications with the ontowiki framework. In T. Pellegrini, S. Auer, K. Tochtermann, & S. Schaffert (Eds.), *Networked Knowledge - Networked Media*, volume 221 of *Studies in Computational Intelligence* (pp. 61–77). Springer Berlin Heidelberg.

[Herlocker et al., 2004] Herlocker, J., Konstan, J., Terveen, L., & Riedl, J. (2004). Evaluating collaborative filtering recommender systems. *ACM Trans. on Information Systems (TOIS)*, 22(1), 53.

[Herlocker et al., 1999] Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *ACM SIGIR conference* (pp. 230–237).

[Hildebrand et al., 2007] Hildebrand, M., van Ossenbruggen, J., & Hardman, L. (2007). *An analysis of search-based user interaction on the Semantic Web*. Technical report, CWI Information Systems.

[Hollenbach et al., 2009] Hollenbach, J., Presbrey, J., & Berners-Lee, T. (2009). Using RDF Metadata to enable Access Control on the Social Semantic Web. In *Workshop on Collaborative Construction, Management and Linking of Structured Knowledge*.

[Hulpuş et al., 2013] Hulpuş, I., Hayes, C., Karnstedt, M., Greene, D., & Jozwowicz, M. (2013). Kanopy: Analysing the semantic network around document topics. In *Machine Learning and Knowledge Discovery in Databases* (pp. 677–680). Springer.

[Huynh et al., 2007] Huynh, D., Karger, D., & Miller, R. (2007). Exhibit: lightweight structured data publishing. In *World Wide Web Conference*.

[Jones, 1972] Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1), 11–21.

[Käfer et al., 2013] Käfer, T., Abdelrahman, A., Umbrich, J., O'Byrne, P., & Hogan, A. (2013). Observing linked data dynamics. *European Semantic Web Conference*.

[Keller et al., 2004] Keller, R. M., Berrios, D. C., Carvalho, R. E., Hall, D. R., Rich, S. J., Sturken, I. B., Swanson, K. J., & Wolfe, S. R. (2004). SemanticOrganizer: A Customizable Semantic Repository for Distributed NASA Project Teams. In *Semantic Web Challenge at the International Semantic Web Conference*.

[Khrouf et al., 2012] Khrouf, H., Milicic, V., & Troncy, R. (2012). Event media. In *Semantic Web Challenge at the International Semantic Web Conference*.

[Kincaid, 2010] Kincaid, J. (2010). EdgeRank: The secret sauce that makes Facebook's news feed tick. TechCrunch.

[Klein & Visser, 2004] Klein, M. & Visser, U. (2004). Guest Editors' Introduction: Semantic Web Challenge 2003. *IEEE Intelligent Systems*, (pp. 31–33).

[Kobsa, 2007] Kobsa, A. (2007). Privacy-enhanced Web Personalization. In *The adaptive web* (pp. 628–670).: Springer.

[Koren, 2008] Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 426–434).: ACM.

[Krötzsch et al., 2006] Krötzsch, M., Vrandečić, D., & Völkel, M. (2006). Semantic MediaWiki. In *International Semantic Web Conference*.

[Lehmann et al., 2014] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., & Bizer, C. (2014). DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*.

[Leskovec et al., 2007] Leskovec, J., Adamic, L. A., & Huberman, B. A. (2007). The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1), 5.

[Linden et al., 2003a] Linden, G., Smith, B., & York, J. (2003a). Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1), 76–80.

[Linden et al., 2003b] Linden, G., Smith, B., & York, J. (2003b). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1), 76–80.

[Liu et al., 2009] Liu, H., Hou, X. Q., Hu, G., Li, J., & Ding, Y. Q. (2009). Development of an EHR System for Sharing - A Semantic Perspective. *Studies in Health Technology and Informatics*, 150, 113–117.

[Loizou, 2009] Loizou, A. (2009). *How to recommend music to film buffs: enabling the provision of recommendations from multiple domains*. PhD thesis, University of Southampton.

[Mandl & Kohane, 2008] Mandl, K. D. & Kohane, I. S. (2008). Tectonic Shifts in the Health Information Economy. *The New England Journal of Medicine*, 358(16), 1732–1737.

[Mangold, 2007] Mangold, C. (2007). A survey and classification of semantic search approaches. *International Journal of Metadata, Semantics and Ontologies*, 2(1), 23–34.

[Mendes et al., 2011] Mendes, P., Jakob, M., García-Silva, A., & Bizer, C. (2011). Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems* (pp. 1–8).: ACM.

[Middleton et al., 2009] Middleton, S. E., Roure, D. D., & Shadbolt, N. R. (2009). Ontology-based recommender systems. In S. Staab & D. Rudi Studer (Eds.), *Handbook on Ontologies*, International Handbooks on Information Systems (pp. 779–796). Springer Berlin Heidelberg. 10.1007/978-3-540-92673-3$_3$5.

[Mika & Akkermans, 2003] Mika, P. & Akkermans, H. (2003). *D1.2 Analysis of the State-of-the-Art in Ontology-based Knowledge Management*. Technical report, SWAP Project.

[Miller et al., 2004] Miller, B., Konstan, J., & Riedl, J. (2004). PocketLens: Toward a Personal Recommender System. *ACM Transactions on Information Systems (TOIS)*, 22(3), 437–476.

[Miller et al., 2003] Miller, B. N., Albert, I., Lam, S. K., Konstan, J. A., & Riedl, J. (2003). Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th international conference on Intelligent user interfaces* (pp. 263–266).: ACM.

[Mitchell-Wong et al., 2007] Mitchell-Wong, J., Kowalczyk, R., Roshelova, A., Joy, B., & Tsai, H. (2007). Opensocial: From social networks to social ecosystem. In *Digital EcoSystems and Technologies Conference, 2007. DEST '07. Inaugural IEEE-IES* (pp. 361–366).

[Mukherjee et al., 2008] Mukherjee, P., Leng, C., & Schurr, A. (2008). Piki - a peer-to-peer based wiki engine. *Peer-to-Peer Computing, IEEE International Conference on*, 0,

185–186.

[Nolin et al., 2008] Nolin, M.-A., Ansell, P., Belleau, F., Idehen, K., Rigault, P., Tourigny, N., Roe, P., Hogan, J. M., & Dumontier, M. (2008). Bio2RDF Network of Linked Data. In *Semantic Web Challenge at the International Semantic Web Conference*.

[Oram, 2001] Oram, A., Ed. (2001). *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. Sebastopol, CA, USA: O'Reilly & Associates, Inc.

[Oren et al., 2007] Oren, E., Haller, A., Hauswirth, M., Heitmann, B., Decker, S., & Mesnage, C. (2007). A flexible integration framework for semantic web 2.0 applications. *Software, IEEE*.

[Oren et al., 2008] Oren, E., Heitmann, B., & Decker, S. (2008). ActiveRDF: embedding Semantic Web data into object-oriented languages. *Journal of Web Semantics*.

[Oren & Tummarello, 2007] Oren, E. & Tummarello, G. (2007). Sindice. In *Semantic Web Challenge at the International Semantic Web Conference*.

[Ostuni et al., 2013] Ostuni, V. C., Di Noia, T., Di Sciascio, E., & Mirizzi, R. (2013). Top-n recommendations from implicit feedback leveraging linked open data. In *Proceedings of the 7th ACM conference on Recommender systems* (pp. 85–92).: ACM.

[Page et al., 1999] Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

[Panchenko, 2011] Panchenko, A. (2011). Comparison of the baseline knowledge-, corpus-, and web-based similarity measures for semantic relations extraction. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics* (pp. 11–21).: Association for Computational Linguistics.

[Passant, 2010a] Passant, A. (2010a). dbrec – music recommendations using dbpedia. *ISWC*.

[Passant, 2010b] Passant, A. (2010b). Measuring semantic distance on linking data and using it for resources recommendations. In *Linked AI: AAAI Spring Symposium" Linked Data Meets Artificial Intelligence", AIII*.

[Passant, 2010c] Passant, A. (2010c). *Semantic Web Technologies For Enterprise 2.0*. IOS Press.

[Passant, 2011] Passant, A. (2011). seevl: mining music connections to bring context, search and discovery to the music you like. In *Semantic Web Challenge at the International Semantic Web Conference*.

[Pazzani & Billsus, 2007] Pazzani, M. & Billsus, D. (2007). Content-based recommendation systems. *The adaptive web*, (pp. 325–341).

[Perugini et al., 2004] Perugini, S., Gonçalves, M., & Fox, E. (2004). Recommender systems research: A connection-centric survey. *Journal of Intelligent Information Systems*, 23(2), 107–143.

[Porcel et al., 2012] Porcel, C., Tejeda-Lorente, A., Martínez, M., & Herrera-Viedma, E. (2012). A hybrid recommender system for the selective dissemination of research resources in a technology transfer office. *Information Sciences*, 184(1), 1–19.

[Quasthoff & Meinel, 2012] Quasthoff, M. & Meinel, C. (2012). Supporting object-oriented programming of semantic-web software. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(1), 15–24.

[Raimond et al., 2008] Raimond, Y., Sandler, M., & Mary, Q. (2008). A web of musical information. In *International Conference on Music Information Retrieval*.

[Resnick et al., 1994] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work* (pp. 175–186).: ACM.

[Rocha et al., 2004] Rocha, C., Schwabe, D., & Aragao, M. P. (2004). A hybrid approach for searching in the semantic web. In *Proceedings of the 13th international conference on World Wide Web* (pp. 374–383).: ACM.

[Sabherwal & Becerra-Fernandez, 2010] Sabherwal, R. & Becerra-Fernandez, I. (2010). *Business Intelligence: Practices, Technologies, & Management*. Wiley.

[Salter & Antonopoulos, 2006] Salter, J. & Antonopoulos, N. (2006). Cinemascreen recommender agent: combining collaborative and content-based filtering. *Intelligent Systems, IEEE*, 21(1), 35–41.

[Schafer et al., 2007] Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative filtering recommender systems. In *The adaptive web* (pp. 291–324). Springer.

[Scharl et al., 2007] Scharl, A., Weichselbraun, A., Hubmann-Haidvogel, A., Stern, H., Wohlgenannt, G., & Zibold, D. (2007). Media Watch on Climate Change: Building and Visualizing Contextualized Information Spaces. In *Semantic Web Challenge at the International Semantic Web Conference*.

[Schein et al., 2002] Schein, A. I., Popescul, A., H., L., Popescul, R., Ungar, L. H., & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *Conference on Research and Development in Information Retrieval* (pp. 253–260).: ACM Press.

[Schreiber et al., 2006] Schreiber, G., Amin, A., van Assem, M., de Boer, V., & Hardman, L. (2006). MultimediaN E-Culture demonstrator. In *Semantic Web Challenge at the International Semantic Web Conference*.

[Shani & Gunawardana, 2011] Shani, G. & Gunawardana, A. (2011). Evaluating recommendation systems. In *Recommender systems handbook* (pp. 257–297). Springer.

[Shinavier, 2010] Shinavier, J. (2010). Optimizing real-time rdf data streams. *CoRR*, abs/1011.3595.

[Sjoberg et al., 2007] Sjoberg, D. I. K., Dyba, T., & Jorgensen, M. (2007). The Future of

Empirical Methods in Software Engineering Research. *Future of Software Engineering*, (pp. 358–378).

[Skaf et al., 2008] Skaf, H., Rahhal, C., & Molli, P. (2008). *Peer-to-peer Semantic wikis*. Research Report RR-6714, INRIA.

[Soni et al., 1995] Soni, D., Nord, R. L., & Hofmeister, C. (1995). Software architecture in industrial applications. *International Conference on Software Engineering*.

[Specia & Motta, 2007] Specia, L. & Motta, E. (2007). Integrating folksonomies with the semantic web. In *The semantic web: research and applications* (pp. 624–639). Springer.

[Story et al., 2009] Story, H., Harbulot, B., Jacobi, I., & Jones, M. (2009). FOAF+SSL: RESTful Authentication for the Social Web. In *Workshop on Trust and Privacy on the Social and Semantic Web*.

[Su & Khoshgoftaar, 2009] Su, X. & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*.

[Trams et al., 2011] Trams, S., Frischmuth, P., Arndt, N., Ermilov, T., & Auer, S. (2011). Waeving a distributed, semantic social network for mobile users. In *Extended Semantic Web Conference*.

[Tran et al., 2007] Tran, T., Haase, P., Lewen, H., Muñoz-García, Ó., Gómez-Pérez, A., & Studer, R. (2007). Lifecycle-Support in Architectures for Ontology-Based Information Systems. In *International Semantic Web Conference*.

[Vargas & Castells, 2011] Vargas, S. & Castells, P. (2011). Rank and relevance in novelty and diversity metrics for recommender systems. In *ACM Conference on Recommender Systems* (pp. 109–116).

[Wang & Kobsa, 2009] Wang, Y. & Kobsa, A. (2009). Technical Solutions for Privacy-Enhanced Personalization. *Intelligent User Interfaces: Adaptation and Personalization Systems and Technologies*.

[Weiss et al., 2007] Weiss, S., Urso, P., & Molli, P. (2007). Wooki: a p2p wiki-based collaborative writing tool. In *Proceedings of the 8th international conference on Web information systems engineering*, WISE'07 (pp. 503–512). Berlin, Heidelberg: Springer-Verlag.

[Wilkinson et al., 2003] Wilkinson, K., Sayers, C., Kuno, H., & Reynolds, D. (2003). Efficient RDF storage and retrieval in Jena2. In *Workshop on Semantic Web and Databases*.

[Winoto & Tang, 2008] Winoto, P. & Tang, T. (2008). If you like the devil wears prada the book, will you also enjoy the devil wears prada the movie? a study of cross-domain recommendations. *New Generation Computing*, 26(3), 209–225.

[Xiao & Benbasat, 2007] Xiao, B. & Benbasat, I. (2007). E-commerce product recommendation agents: Use, characteristics, and impact. *Management Information Systems Quarterly*, 31(1), 137.

[Xu et al., 2005] Xu, R., Wunsch, D., et al. (2005). Survey of clustering algorithms. *Neural*

Networks, IEEE Transactions on, 16(3), 645–678.

[Yegnanarayana, 2009] Yegnanarayana, B. (2009). *Artificial neural networks*. PHI Learning Pvt. Ltd.