Recommender Systems for Social Bookmarking

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Tilburg,
op gezag van de rector magnificus,
prof. dr. Ph. Eijlander,
in het openbaar te verdedigen ten overstaan van een
door het college voor promoties aangewezen commissie
in de aula van de Universiteit
op dinsdag 8 december 2009 om 14.15 uur

door

Antonius Marinus Bogers,
geboren op 21 september 1979 te Roosendaal en Nispen

Promotor:

Prof. dr. A.P.J. van den Bosch

Beoordelingscommissie:

Prof. dr. H.J. van den Herik

Prof. dr. M. de Rijke

Prof. dr. L. Boves

Dr. B. Larsen

Dr. J.J. Paijmans

The research reported in this thesis has been funded by SenterNovem / the Dutch Ministry of Economic Affairs as part of the IOP-MMI À Propos project.

SIKS Dissertation Series No. 2009-42

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

TiCC Dissertation Series No. 10

ISBN 978-90-8559-582-3

> *The Web, they say, is leaving the era of search and entering one of discovery. What's the difference? Search is what you do when you're looking for something. Discovery is when something wonderful that you didn't know existed, or didn't know how to ask for, finds you.*

Jeffrey M. O'Brien

Preface

First and foremost I would like to thank my supervisor and promotor Antal van den Bosch, who guided me in my first steps as a researcher, both for my Master's thesis and my Ph.D. research. Antal always gave me free reign in investigating many different research problems, while at the same time managing to steer me in the right direction when the time called for it. Antal was always able to make time for me or any of the other Ph.D. students, and read and comment on paper or presentation drafts.

In addition to turning me into a better researcher, Antal was also instrumental in improving my Guitar Hero skills. Our thesis meetings during your sabbatical doubled as a kind of Rock 'n Roll Fantasy Camp, where we could both unwind from discussing yet another batch of experiments I had run or was planning to run. Rock on! Antal also shares my passion for ice hockey. This resulted in us attending Tilburg Trappers games in Stappegoor as well as our regular discussions of the latest hockey news. Thanks for inviting me to come see the NHL All Star games in Breda. Hopefully we will meet again in spirit come May 2010 when the Canucks beat the Penguins in the Stanley Cup finals!

The research presented in this thesis was performed in the context of the À Propos project. I would like to acknowledge SenterNovem and the Dutch Ministry of Economic Affairs for funding this project as part of the IOP-MMI program. The À Propos project was started by Lou Boves, Antal, and Frank Hofstede. I would like to thank Lou and Frank in particular. Frank was always able to look at my research problems from a different and more practical angle, and as a result our discussions were always very stimulating. I would also like to Mari Carmen Puerta-Melguizo, Anita Deshpande, and Els den Os, as well as the other members and attendees of the project meetings for the pleasant cooperation and helpful comments and suggestions.

I wish to thank the members of my committee for taking time out of their busy schedules to read my dissertation and attending my defense: Jaap van den Herik, Maarten de Rijke, Lou Boves, Birger Larsen, and Hans Paijmans. Special thanks go to Jaap for his willingness to go through my thesis with a fine-grained comb. The readability of the final text has benefited greatly from his meticulous attention to detail and quality. Any errors remaining in the thesis are my own. I would also like to thank Birger for his comments, which helped to dot the i's and cross the t's of the final product. Finally, I would like to thank Hans Paijmans, who contributed considerably to my knowledge of IR.

My Ph.D. years would not have been as enjoyable and successful without my colleagues at Tilburg University, especially those at the ILK group. It is not everywhere that the bond between colleagues is as strong as it was in ILK and I will not soon forget the coffee breaks with the Sulawesi Boys, the BBQs and Guitar Hero parties, lunch runs, after-work drinks, and the friendly and supportive atmosphere on the 3rd floor of Dante. I do not have enough room to thank everyone personally here, you know who you are. In your own way, you all contributed to this thesis.

Over the course of my Ph.D. I have spent many Fridays at the Science Park in Amsterdam, working with members of the ILPS group headed by Maarten de Rijke. I would like to thank Erik Tjong Kim Sang for setting this up and Maarten for allowing me to become a guest researcher at his group. Much of what I know about doing IR research, I learned from these visits. From small things like visualizing research results and LaTeX layout to IR research methodology and a focus on empirical, task-driven research. I hope that some of what I have learned shows in the thesis. I would like to thank all of the ILPS members, but especially Krisztian, Katja, and Maarten for collaborating with me on expert search, which has proven to be a very fruitful collaboration so far.

I have also had the pleasure of working at the Royal School of Library and Information Science in Copenhagen. I am most grateful to Birger Larsen and Peter Ingwersen, for helping to arrange my visit and guiding me around. Thanks are also due to Mette, Haakon, Charles, Jette, and the other members of the IIIA group for welcoming me and making me feel at home. Jeg glæder mig til at arbejde sammen med jer snart.

Thanks are due to Sunil Patel for designing part of the stylesheet of this thesis and to Jonathan Feinberg of http://www.wordle.net/ for the word cloud on the front of this thesis. I owe Maarten Clements a debt of gratitude for helping me to more efficiently implement his random walk algorithm. And of course thanks to BibSonomy, CiteULike, and Delicious for making the research described in this thesis possible.

Finally, I would like to thank the three most important groups of people in my life. My friends, for always supporting me and taking my mind off my work. Thanks for all the dinners, late-night movies, pool games, talks, vacations and trips we have had so far! Thanks to my parents for always supporting me and believing in me; without you I would not have been where I am today. Kirstine, thanks for putting up with me while I was distracted by my work, and thanks for patiently reading and commenting on my Ph.D. thesis. Og tusind tak fordi du bringer så meget glæde, latter og kærlighed ind i mit liv. Det her er til Timmy og Oinky!

Contents

Chapter 1

Introduction

For the past two decades, the World Wide Web has expanded at enormous rate. The first generation of the World Wide Web (WWW) enabled users to have instantaneous access to a large diversity of knowledge items. The second generation of the WWW is usually denoted by *Web 2.0*. It signifies a fundamental change in the way people interact with and through the World Wide Web. Web 2.0 is also referred to as the participatory Web. It can be characterized as a paradigm that facilitates communication, interoperability, user-centered design, and information sharing and collaboration on the Web (O'Reilly, 2005; Sharma, 2008). Moreover, in the transition to Web 2.0 we see a paradigm shift from local and solitary to global and collaborative. Also, this shift coincides with a shift from accessing and creating information to understanding information and understanding the people who deal with this information. Instead of creating, storing, managing, and accessing information on only one specific computer or browser, information management and access has been moving to many distributed places on the Web. Collaboratively created websites such as Wikipedia are edited and accessed by anyone, and users can document and share any aspect of their lives online using blogs, social networking sites, and video and photo sharing sites.

This thesis deals with recommender systems, social tagging, and social bookmarking. What are the relations between these three elements, and can we build recommender systems that profit from the presence of the other two elements? Assuming that we can, what are the threats from the outside or inside of this new part of the WWW? In the thesis we deal with spam as the outside threat, and duplicates as the inside threat. The aim of the thesis is to understand the symbiosis of recommender systems, social tagging, and social bookmarking, and to design mechanisms that successfully counter the threats from the outside and from the inside.

The course of this chapter is as follows. We introduce social bookmarking in Section 1.1. It is followed by a description of the scope of the thesis. The problem statement and five research questions are formulated in Section 1.3. Section 1.4 describes the research methodology. The structure of the thesis is provided in Section 1.5. Finally, Section 1.6 points to the origins of the material.

1.1 Social Bookmarking

Social bookmarking is a rather new phenomenon: instead of keeping a local copy of pointers to favorite URLs, users can instead store and access their bookmarks online through a Web interface. The underlying application then makes all stored information shareable among users. Closely related to social bookmarking websites are the so-called *social reference managers*, which follow the same principle, but with a focus on the online management and access of scientific articles and papers. Social bookmarking websites have seen a rapid growth in popularity and a high degree of activity by their users. For instance, Delicious[1] is one of the most popular social bookmarking services. It received an average of 140,000 posts per day in 2008 according to the independently sampled data collected by Philipp Keller[2]. In addition to the aforementioned functionality, most social 'storage' services also offer the user the opportunity to describe by keywords the content they added to their personal profile. These keywords are commonly referred to as *tags*. They are an addition to e.g., the title and summary metadata commonly used to annotate content, and to improve the access and retrievability of a user's own bookmarked Web pages. These tags are then made available to all users, many of whom have annotated many of the same Web pages with possibly overlapping tags. This results in a rich network of users, bookmarks, and tags, commonly referred to as a *folksonomy*. This *social tagging* phenomenon and the resulting folksonomies have become a staple of many Web 2.0 websites and services (Golder and Huberman, 2006).

The emerging folksonomy on a social bookmarking website can be used to enhance a variety of tasks, such as searching for specific content. It can also enable the active discovery of new content by allowing users to browse through the richly connected network. A user could select one of his[3] tags to explore all bookmarks annotated with that tag by the other users in the network, or locate like-minded users by examining a list of all other users who added a particular bookmark, possibly resulting in serendipitously discovered content (Marlow et al., 2006). Both browsing and searching the system, however, require active user participation to locate new and interesting content. As the system increases in popularity and more users as well as content enter the system, the access methods become less effective at finding all the interesting content present in the system. The information overload problem caused by this growing influx of users and content means that search and browsing, which require active participation, are not always the most practical or preferable ways of locating new and interesting content. Typically, users only have a limited amount of time to go through the search results. Assuming users know about the existence of the relevant content and know how to formulate the appropriate queries they may arrive in time at the preferred places. But what happens when the search and browse process becomes less effective? And what if the user does *not* know about all relevant content available in the system? Our interest, and the focus of this thesis, lies in using *recommender systems* to help the user with this information overload problem, and automatically find interesting content for the user. A recommender system is a type of personalized information filtering technology used to identify sets of items that are likely to be of interest to a certain user, using a variety of

[1] http://www.delicious.com/

[2] Available at http://deli.ckoma.net/stats; last visited January 2009.

[3] In this thesis, we use 'his' and 'he' to refer to both genders.

information sources related to both the user and the content items (Resnick and Varian, 1997).

1.2 Scope of the Thesis

In this thesis, we investigate how recommender systems can be applied to the domain of social bookmarking. More specifically, we want to investigate the task of *item recommendation*. For this purpose, interesting and relevant items—bookmarks or scientific articles—are retrieved and recommended to the user. Recommendations can be based on a variety of information sources about the user and the items. It is a difficult task as we are trying to predict which items out of a very large pool would be relevant given a user's interests, as represented by the items which the user has added in the past. In our experiments we distinguish between two types of information sources. The first one is *usage data* contained in the folksonomy, which represents the past selections and transactions of all users, i.e., who added which items, and with what tags. The second information source is the *metadata* describing the bookmarks or articles on a social bookmarking website, such as title, description, authorship, tags, and temporal and publication-related metadata. We are among the first to investigate this content-based aspect of recommendation for social bookmarking websites. We compare and combine the content-based aspect with the more common usage-based approaches.

Because of the novelty of applying recommender systems to social bookmarking websites, there is not a large body of related work, results, and design principles to build on. We therefore take a system-based approach for the evaluation our work. We try to simulate, as realistically as possible, the reaction of the user to different variants of the recommendation algorithms in a controlled laboratory setting. We focus on two specific domains: (1) recommending bookmarks of Web pages and (2) recommending bookmarked references to scientific articles. It is important to remark, however, that a system-based evaluation can only provide us with a provisional estimate of how well our algorithms are doing. User satisfaction is influenced by more than just recommendation accuracy (Herlocker et al., 2004) and it would be essential to follow up our work with an evaluation on real users in realistic situations. However, this is not the focus of the thesis, nor will we focus on tasks such as tag recommendation or finding like-minded users. We focus strictly on recommending items.

1.3 Problem Statement and Research Questions

As stated above, the rich information contained in social bookmarking websites can be used to support a variety of tasks. We consider three important ones: browsing, search, and recommendation. From these three, we focus on (item) recommendation in this thesis. In this context we may identify two types of key characteristics of social bookmarking websites that can be used in the recommendation process. We remark that the information sources represented by these characteristics are not always simultaneously available in every recommendation scenario. The resulting recommendations are produced by (1) collaborative

filtering algorithms and (2) content-based filtering algorithms. We briefly discuss both types of algorithms and the associated characteristics below.

Collaborative filtering algorithms Much of the research in recommender systems has focused on exploiting sets of usage patterns that represent user preferences and transactions. The class of algorithms that operate on this source of information are called *Collaborative Filtering* (CF) algorithms. They automate the process of "word-of-mouth" recommendation: items are recommended to a user based on how like-minded users rated those items (Goldberg et al., 1992; Shardanand and Maes, 1995). In the social bookmarking domain, we have an extra layer of usage data at our disposal in the folksonomy in the form of tags. This extra layer of collaboratively generated tags binds the users and items of a system together in yet another way, opening up many possibilities for new algorithms that can take advantage of this data.

Content-based filtering algorithms Social bookmarking services and especially social reference managers are also characterized by the rich metadata describing the content added by their users. Recommendation on the basis of textual information is commonly referred to as *content-based filtering* (Goldberg et al., 1992) and matches the item metadata against a representation of the user's interest to produce new recommendations. The metadata available on social bookmarking services describe many different aspects of the items posted to the website. It may comprise both personal information, such as reviews and descriptions, as well as general metadata that is the same for all users. While the availability of metadata is not unique to social bookmarking—movie recommenders, for instance, also have a rich set of metadata at their disposal (Paulson and Tzanavari, 2003)—it might be an important information source for generating item recommendations.

Having distinguished the two types of characteristics of social bookmarking websites, we are now able to formulate our problem statement (PS).

> **PS** *How can the characteristics of social bookmarking websites be exploited to produce the best possible item recommendations for users?*

To address this problem statement, we formulate five research questions. The first two research questions belong together. They read as follows.

> **RQ 1** How can we use the information represented by the folksonomy to support and improve the recommendation performance?
>
> **RQ 2** How can we use the item metadata available in social bookmarking systems to provide accurate recommendations to users?

After answering the first two questions, we know how to exploit in the best manner the two types of information sources—the folksonomy and item metadata—to produce accurate recommendations. This leads us to our third research question.

RQ 3 Can we improve performance by combining the recommendations generated by different algorithms?

These are the three main research questions. As mentioned earlier, we evaluate our answers to these questions by simulating the user's interaction with our proposed recommendation algorithms in a laboratory setting. However, such an idealized perspective does not take into account the dynamic growth issues caused by the increasing popularity of social bookmarking websites. Therefore, we focus on two of these growing pains. There is one pain attacking social bookmarking websites from the outside, spam. The other one, duplicate content, attacks a social bookmarking website from the inside. They lead to our final two research questions.

RQ 4 How big a problem is spam for social bookmarking services?

RQ 5 How big a problem is the entry of duplicate content for social bookmarking services?

Wherever it is applicable and aids our investigation, we will break down these questions into separate and even more specific research questions.

1.4 Research Methodology

The research methodology followed in the thesis comprises five parts: (1) reviewing the literature, (2) analyzing the findings, (3) designing the recommendation algorithms, (4) evaluating the algorithms, and (5) designing protection mechanisms for two growing pains. First, we conduct a literature review to identify the main techniques, characteristics, and issues in the fields of recommender systems, social tagging, and social bookmarking, and in the intersection of the three fields. In addition, Chapters 4 through 8 each contain short literature reviews specifically related to the work described in the respective chapters.

Second, we analyze the findings from the literature. We use these in the third part of our methodology to guide us in the development of recommendation algorithms specifically suited for item recommendation on social bookmarking websites.

Fourth, we evaluate our recommendation algorithms in a quantitative manner. The building blocks of our quantitative evaluation are described in more detail in Chapter 3. Our quantitative evaluation is based on a so-called *backtesting* approach to evaluation that is common in recommender systems literature (Breese et al., 1998; Herlocker et al., 2004; Baluja et al., 2008). In backtesting, we evaluate on a per-user basis. We withhold randomly selected items from each user profile, and generate recommendations by using the remaining data as training material. If a user's withheld items are predicted at the top of the ranked list of recommendations, then the algorithm is considered to perform well for that user. The performance of a recommendation algorithm is averaged over the performance for all individual users. In our evaluation we employ cross-validation, which can provide a

reliable estimate of the true generalization error of our recommendation algorithms (Weiss and Kulikowski, 1991). Our experimental setup is described in more detail in Section 3.4.

The fifth and final part of our research methodology involves designing protection for social bookmarking websites and our proposed recommendation algorithms against two growing pains: spam and duplicate content. For both pains we analyze how extensive the problem is for one of our data sets. We then design algorithms to automatically detect this problematic content. Finally, we perform a robustness analysis of the recommendation algorithms we proposed in Part I against spam and duplicates.

1.5 Organization of the Thesis

The thesis consists of two main parts. The first part is 'Recommending Bookmarks', which ranges from Chapter 3 to Chapter 6, both inclusive. The second main part is 'Growing Pains', which covers Chapters 7 and 8. The two Parts are preceded by two introductory chapters. Chapter 1 contains an introduction of the thesis as well as the formulation of a problem statement and five research questions. Moreover, the research methodology is given. Chapter 2 contains a literature review. Below we provide a brief overview of the contents of Parts I and II.

Part I The core of our recommendation experiments is contained in Part I. It starts in Chapter 3 with an overview of the building blocks of our quantitative evaluation. We start by formally defining the recommendation task we are trying to solve: recommending interesting items to users based on their preferences. We then introduce our data sets and describe how they were collected. We discuss our experimental setup, from data pre-processing and filtering to our choice of evaluation metrics.

Then, in Chapter 4 we investigate the first of two important characteristics of social bookmarking systems: the presence of the folksonomy. We propose and compare different options for using the folksonomy in item recommendation for social bookmarking (RQ 1).

In Chapter 5, we investigate the usefulness of item metadata in the recommendation process, the other characteristic of social bookmarking websites, and examine how we can use this metadata to improve recommendation performance (RQ 2).

Chapter 6 concludes Part I by examining different options for combining these two characteristics to see if we can improve upon our best-performing algorithms from Chapters 4 and 5 by combining the output of the different algorithms into a new list of recommendations (RQ 3).

Part II In Part II, we dive into the periphery of recommendation, and zoom in on two specific growing pains that social bookmarking services encounter as they become more popular: spam and duplicate content. In Chapter 7 we take a look at the problem of spam in social bookmarking websites to attempt to quantify the problem. We propose a spam detection algorithm based on a combination of language models and the

nearest-neighbor algorithm. We also investigate the influence spam can have on recommending items for social bookmarking websites in a case study on one of our data sets (RQ 4).

In Chapter 8 we take a similar approach to the problem of duplicate content. We start by quantifying the problem and then construct a classifier that can automatically identify duplicate item pairs in one of our data sets. Finally, we investigate the influence of duplicates on recommendation performance in a case study on one of our data sets.

Part III concludes the thesis. We revisit our research questions and the answers we found. Then we answer the problem statement and formulate our conclusions. We also list future research directions, drawing on the work described in this thesis.

Additional information that would interrupt the narrative is placed in Appendices A, D, and C, and referred to in the text where applicable. We also list some mark-up conventions here. Tags feature prominently in our thesis; for clarity we print them with a sans-serif font, e.g., as information-retrieval or toread. We print metadata fields with a fixed-width font, e.g., as `TITLE` or `ABSTRACT`. We experiment with many different combinations of algorithms and similarity metrics, each resulting in a set of recommendations for a group of test users. We will refer to such output as a recommendation *run* which is made up of a list of recommendations. Different variants of an algorithm are marked up as RunName1 or RunName2 where it helps to clarify the discussion.

1.6 Origins of the Material

This thesis is partly based on papers that have already been published or have been submitted. Early versions of the experiments with the CiteULike data set in Part I were described in Bogers and Van den Bosch (2008a), and later expanded to include more algorithms and more data sets in Bogers and Van den Bosch (2009b). In an earlier stage, the foundations of the work on content-based recommendation were laid in Bogers and Van den Bosch (2007). Finally, the work on spam detection described in Part II was published first in Bogers and Van den Bosch (2008b) and in expanded form in Bogers and Van den Bosch (2009a).

Chapter 2

Related Work

The work presented in this thesis is novel in (1) its application of recommendation algorithms to social bookmarking websites, and (2) its incorporation of the information represented by the folksonomy and the metadata on those websites. This chapter serves as an introduction into general related work covering some subareas of our work. All related work specifically relevant to the work described in each of the following chapters will be discussed in those respective chapters.

We start this chapter in Section 2.1 by introducing recommender systems: first, a brief history of the field will be given, followed by the most popular algorithms and applications, as well as the most common problems that the recommender systems are suffering. Then, we take a more detailed look at related work on recommending Web bookmarks and references to scientific articles. We briefly discuss the role of the user and context in recommendation. In Section 2.2 we closely consider the phenomenon of social tagging which has become a big part of the Web 2.0 paradigm. We compare it to the traditional view on indexing by information specialists and we discuss two different types of social tagging. We establish that social bookmarking services are a class of social websites that lean heavily on social tagging for managing their content. Section 2.3 discusses the rise of social bookmarking services and their characteristics, as well as research into the different user tasks performed on social bookmarking websites.

2.1 Recommender Systems

The explosive growth of the World Wide Web in the 1990s resulted in a commensurate growth of the amount of information available online, outgrowing the capacity of individual users to process all this information. This prompted a strong interest in the specific research fields and technology that could help manage this information overload. The most characteristic fields are *information retrieval* and *information filtering*. As a research field, information retrieval (IR) originated in the 1950s and is concerned with automatically matching a user's information need against a collection of documents. The 1990s saw a change

in focus from small document collections to the larger collections of realistic size needed to cope with the ever-growing amount of information on the Web. Information filtering (IF) systems aim to help the users make optimal use of their limited reading time by filtering out unwanted information in order to expose only the relevant information from a large flow of information, such as news articles (Hanani et al., 2001). Typically, such systems construct a model of a user's interests and match that against the incoming information objects. While IR and IF are considered separate research fields, they share many characteristics, such as a focus on analyzing textual content (Belkin and Croft, 1992).

A third type of technology designed to combat information overload are recommender systems, which have their origin in the field of information filtering (Hanani et al., 2001). A recommender system is used to identify sets of items that are likely to be of interest to a certain user, exploiting a variety of information sources related to both the user and the content items. In contrast to information filtering, recommender systems actively predict which items the user might be interested in and *add* them to the information flowing to the user, whereas information filtering technology is aimed at *removing* items from the information stream (Hanani et al., 2001). Over the past two decades many different recommendation algorithms have been proposed for many different domains. In Subsections 2.1.1 through 2.1.3, we discuss the three most common classes of algorithms: collaborative filtering, content-based filtering, and knowledge-based recommendation. We explain how the algorithms work, and discuss the most important advantages and problems of each algorithm. Then, in Subsection 2.1.4, we discuss related work on recommendation Web pages and scientific articles. We conclude this section by briefly discussing the role of the user and context in the recommendation process in Subsection 2.1.5.

Finally, we would like to remark that recommender systems technology has also been applied to a variety of different domains, such as online stores (Linden et al., 2003), movies (Herlocker et al., 1999), music (Celma, 2008), Web pages (Joachims et al., 1997), e-mail (Goldberg et al., 1992), books (Mooney and Roy, 2000), news articles (Das et al., 2007), scientific articles (Budzik and Hammond, 1999), and even jokes (Goldberg et al., 2001). In October 2006, it also spawned a million dollar competition in the form of the Netflix Prize[1], aimed at designing a recommender system that significantly outperforms Netflix's own CineMatch system. We refer the reader to Montaner et al. (2003) for a more comprehensive overview of recommender systems and domains.

2.1.1 Collaborative Filtering

Arguably the most popular class of recommendation algorithms, *collaborative filtering* (CF), descend from work in the area of information filtering. CF algorithms try to automate the process of "word-of-mouth" recommendation: items are recommended to a user based on how like-minded users rated those items (Shardanand and Maes, 1995). The term 'collaborative filtering' was first used by Goldberg et al. (1992), to describe their Tapestry filtering system, which allowed users to annotate documents and e-mail messages. Other users could then request documents annotated by certain people, but identifying these people was left

[1] http://www.netflixprize.com/

to the user. Subsequent CF approaches automated this process of locating the nearest neighbors of the active user. Important early work was done by Resnick et al. (1994) on their GROUPLENS system for recommending Usenet articles, and by Shardanand and Maes (1995) on their RINGO music recommender system. They were the first to correlate the rating behavior between users in order to (1) determine the most similar neighbors and (2) use them to predict interest in new items. This work was expanded upon by Breese et al. (1998) and Herlocker et al. (1999), who performed the first large-scale evaluation and optimization of collaborative filtering algorithms. Herlocker et al. (2004) also published important work on evaluating recommender systems, which we adhere to in our experimental setup described in Chapter 3.

CF algorithms exploit set of usage patterns that represent user preferences and transactions to match them with those of people who share the same tastes or information needs. After locating a possible match, the algorithm then generate the recommendations. The preference patterns can be directly elicited from the user. A telling example is the Amazon website[2] where customers are asked to rate an item on a scale from 1 to 5 stars. The patterns can also be inferred implicitly from user actions, such as purchases, reading time, or downloads. After gathering the user opinions, either explicitly or implicitly, they are commonly represented in a user-item ratings matrix, such as the ones shown in Figure 2.1. The majority of the cells in these matrix are empty, because it is usually impossible for a user to select, rate, or purchase all items in the system. CF algorithms operate on such a user-item matrix to predict values for the empty entries in the matrix.

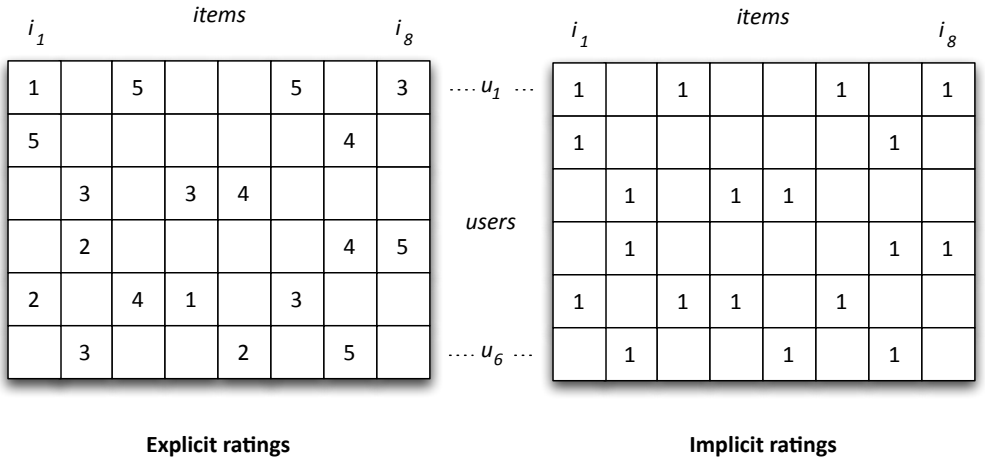


Figure 2.1: Two examples of user-item matrices for a toy data set of six users and eight items. Values in the matrix can be ratings in the case of explicit user opinions (left), or unary in the case of implicit user activity (right).

CF algorithms are commonly divided into two types, *memory-based algorithms* and *model-based algorithms*, analogous to the way machine learning algorithms can be categorized into two classes.

[2] http://www.amazon.com/

Memory-based Collaborative Filtering Memory-based algorithms are also known as *lazy* recommendation algorithms, because they defer the actual computational effort of predicting a user's interest in an item to the moment a user requests a set of recommendations. The training phase of a memory-based algorithm consists of simply storing all the user ratings into memory. There are two variants of memory-based recommendation and both are based on the k-Nearest Neighbor algorithm from the field of machine learning (Aha et al., 1991): *user-based filtering* and *item-based filtering*.

In user-based filtering, the active user is matched against the ratings matrix to find the neighboring users with which the active user has a history of agreeing. This is typically done using metrics such as Pearson's correlation coefficient or the cosine similarity. Once this neighborhood has been identified, all items in the neighbors' profiles unknown to the active user are considered as possible recommendations and sorted by their frequency in that neighborhood. A weighted aggregate of these frequencies is used to generate the recommendations (Herlocker et al., 1999). Item-based filtering was proposed by Sarwar et al. (2001) and is the algorithm of choice of the online store Amazon (Linden et al., 2003). It focuses on finding the most similar items instead of the most similar users. As in user-based filtering, similarities between item pairs are calculated using Pearson's correlation coefficient or the cosine similarity (Breese et al., 1998). Items are considered to be similar when the same set of users has purchased them or rated them highly. For each item of an active user, the neighborhood of most similar items is identified. Each of the top k neighbors is placed on a candidate list along with its similarity to the active user's item. Similarity scores of items occurring multiple times in the candidate list are added. The candidate list is sorted on these aggregated similarity scores and the top N recommendations are then presented to the user (Sarwar et al., 2001; Karypis, 2001). We explain both algorithms in more detail in Section 4.3.

Model-based Collaborative Filtering Also known as *eager* recommendation algorithms, model-based CF algorithms do most of the hard work in the training phase, where they construct a predictive model of the recommendation problem. Generating the recommendations is then a quick and straightforward matter of applying the derived model[3]. Many different machine learning algorithms have been applied in recommender systems, such as Naive Bayes (Breese et al., 1998) and rule induction (Basu et al., 1998), with more emphasis on latent factor models in the past decade. Latent factor models try to reduce the dimensionality of the space of user-item ratings by mapping users and items to the same latent factor space (Koren, 2008). The users and items are then related to each other through these latent factors. The factors can range in ease of interpretation from intuitive, such as movie genres or 'amount of plot twists', to less well defined dimensions, such as 'quirky characters', or even completely uninterpretable dimensions of the user-item relation (Koren, 2008). Examples of latent factor techniques applied to recommendation include Singular Value Decomposition (SVD) by Sarwar et al. (2002), factor analysis by Canny (2002), Probabilistic Latent Semantic Analysis (PLSA) by Hofmann (2004), and non-negative matrix factorization by Koren (2008).

[3]Note that when the similarity computations are pre-computed in, for instance, a nightly cycle, the user-based and item-based filtering algorithms both turn into eager recommendation algorithms.

Advantages and Problems of Collaborative Filtering CF algorithms have several advantages, such as being able to take the quality of an item—or any lack thereof—into account when recommending items, especially in the case of explicit user ratings. For instance, a local band might fall in the same musical genre as an internationally renowned rock band, but this does not guarantee that they are of the same quality. This shows that recognizing the quality of items is clear advantage of CF. By taking actual user preferences into account, CF algorithms can prevent poor recommendations. A second advantage is that CF algorithms are especially useful in domains where content analysis is difficult or costly, such as movie and music recommendation, without requiring any domain knowledge (Burke, 2002).

While the quality of CF algorithms tends to improve over time, the biggest problem is the startup phase of the recommender system, when there are already many items in the system, but few users and no ratings. This is commonly referred to as the *cold-start* problem and means the recommender system cannot generate any recommendations (Schein et al., 2002). Solutions to this problem include using other data sets to seed the system, and using different recommendation algorithms in this startup phase that do not suffer from this problem. Even after acquiring more ratings from the users, sparsity of the user-item matrix can still be a problem for CF. A second problem is what is referred to as the 'gray sheep' problem according to Claypool et al. (1999), which describes the difficulty of recommending for people who are not part of a clear group. Collaborative recommenders work best for user who fit into a specific niche with many similar neighbors (Burke, 1999).

2.1.2 Content-based Filtering

Content-based recommendation algorithms, also known as *content-based filtering*, form the second popular class of algorithms. They can be seen as an extension of the work done on information filtering (Hanani et al., 2001). Typically, content-based filtering approaches focus on building some kind of representation of the content in a system and then learning a profile of a user's interests. The content representations are then matched against the user's profile to find the items that are most relevant to that user. As with CF, the representations of the user profiles are long-term models, and updated as more preference information becomes available (Burke, 2002). Usually, content-based filtering for recommendation is approached as either an IR problem, where document representations have to be matched to user representations on textual similarity; or as a machine learning problem, where the textual content of the representations are incorporated as feature vectors, which are used to train a prediction algorithm. Examples of the IR approach include Whitman and Lawrence (2002) and Bogers and Van den Bosch (2007); examples of the machine learning point of view include Lang (1995) and Mooney and Roy (2000). In Chapter 5 we propose different content-based recommendation algorithms to incorporate the metadata present in social bookmarking systems, so we refer to Section 5.4 for a more extensive discussion of related work in content-based filtering.

Advantages and Problems of Content-based Filtering A clear advantage of content-based filtering algorithms is that they do not require domain knowledge, and that it is sufficient to collect implicit feedback from the users about their item preferences. This can make

content-based filtering the preferred algorithm in domains where eliciting explicit ratings from users is difficult or cumbersome, and where domain knowledge is hard to come by. A second advantage is that content-based filtering algorithms are better at finding topically similar items than CF algorithms because of their explicit focus on textual similarity. However, this can be a disadvantage in domains where content analysis is difficult or impractical to do in large numbers, such as movies and music. Content-based filtering algorithms also tend to get stuck in a 'well of similarity' (Rashid et al., 2002), where they can only recommend items from a narrow topic range; serendipitous recommendations can therefore be hard to achieve.

2.1.3 Knowledge-based Recommendation

All personalized recommendation algorithms attempt to infer which items a user might like. Collaborative filtering algorithms do this based on the behavior of the user and other like-minded users, whereas content-based filtering approaches do this based on the textual representations of the user's interests and the available items. A third class of recommendation algorithms is formed by *knowledge-based algorithms*. They use rules and patterns, and recommend items based on functional knowledge of how a specific item meets a particular user need (Burke, 2002). Such techniques allow the algorithm to reason about the relationship between a user and the available items. This can prevent a recommender system from generating useless recommendations. An example of such a useless recommendation would be recommending milk to a supermarket shopper: the odds of buying milk are so high that milk will always be correlated with everything else in a user's shopping basket, and thus always recommended to the user. Because a knowledge-based recommender system knows what foods ought to go together, it can screen out such useless suggestions (Burke, 1999).

Knowledge-based recommender systems often borrow techniques from the field of *case-based reasoning* (CBR), which is useful for solving constraint-based problems such as the 'milk' problem. In CBR, users can specify content-based attributes which limit the returned recommendation set. Old problem-solving cases are stored by the CBR system, and new situations are then compared against these old cases with the most similar cases being used for solving the new problem (Burke, 1999; McNee, 2006). Recommender systems using such techniques support rating items on multiple dimensions. An example is the ENTREE restaurant recommender system developed by Burke et al. (1997). ENTREE allows restaurants to be rated on price, food quality, atmosphere, service, and other dimensions.

Advantages and Problems of Knowledge-based Recommendation Rating content on multiple dimensions allows the user to provide a rich specification of his recommendation need, which in turn results in more satisfying recommendations. An second advantage of knowledge-based recommendation is that it does not suffer from the cold start problem, and that it allows for intuitive explanations of why a certain item, such as a restaurant, was recommended. In addition to the ENTREE recommender system, other examples of knowledge-based recommender systems are given by Schmitt and Bergmann (1999), Towle and Quinn (2000), and Van Setten et al. (2004). The biggest problem of knowledge-based algorithms is the need for an explicit knowledge acquisition phase (Burke, 2002), which is

difficult in domains without a rich set of attributes. As a result, knowledge-based recommendation is not as popular as the other two classes of algorithms. We do not consider knowledge-based algorithms for our recommendation experiments because we would run into this knowledge acquisition bottleneck when porting our algorithms from one domain to another. Instead, we take a data-driven approach and restrict ourselves to CF and content-based filtering in Chapters 4 and 5 because they match the two characteristic information sources available on social bookmarking services: the folksonomy and metadata.

2.1.4 Recommending Bookmarks & References

In this thesis, we focus on recommendation for social bookmarking websites covering two different domains: Web pages and scientific articles. While there is little related work on item recommendation for social bookmarking, there have been plenty of stand-alone approaches to recommending Web pages and scientific articles. Most of these are focused around the concept of an *Information Management Assistant* (or IMA), that locates and recommends relevant information for the user by inferring a model of the user's interests (Budzik and Hammond, 1999). One of the earliest examples of such a personal information agent was the Memex system envisioned by Vannevar Bush, which was a "device in which an individual stores all his books, records, and communications" and offered the possibility of associative links between information, although it lacked automatic suggestion of related information (Bush, 1945).

The first real IMAs started appearing in the 1990s and mostly focused on tracking the user's browsing behavior to automatically recommend interesting, new Web pages. LETIZIA was among the first of such pro-active IMAs (Lieberman, 1995), and used a breadth-first search strategy to follow, evaluate, and recommend outgoing links from pages in which the user previously showed an interest. In an IMA scenario, strong user interest in a page is typically inferred using heuristics such as a long dwell time on the page, revisiting it several times, or saving or printing the page. Many other IMAs for recommending Web pages have been proposed since then, among which the SYSKILL & WEBERT agent by Pazzani et al. (1996), which allowed users to rate Web pages they visited. It extracted the key terms from those favorite Web pages, and used those to generate queries to be sent to search engines. The search results were then presented to the user as recommendations. Other examples of IMAs that support Web browsing include LIRA by Balabanovic (1998), WEBWATCHER by Joachims et al. (1997), WEBMATE by Chen and Sycara (1998), and the work by Chirita et al. (2006). The GIVEALINK system by Stoilova et al. (2005) is also worth mentioning because of its similarity to social bookmarking: GIVEALINK[4] is a website that asks users to donate their bookmarks files, effectively creating a social bookmarking website. There are some differences with social bookmarking as we describe it in Section 2.3 though: a user's bookmark profile in GIVEALINK is static; a user can only update their bookmarks by re-uploading the entire bookmarks file. Also, GIVEALINK does not support tagging of bookmarks. The bookmarks donated by users are used in tasks such as (personalized) search and recommendation.

[4] http://www.givealink.org/

In addition to Web browsing, IMAs have also been used to support other information-related activities such as writing research papers. Budzik and Hammond (1999) designed an IMA called WATSON that observed user interaction with a small range of everyday applications (e.g., Netscape Navigator, Microsoft Internet Explorer, and Microsoft Word). They constructed queries based on keywords extracted from the documents or Web pages being viewed or edited by the user and sent those queries to the search engines. They report that a user study showed that 80% of the participants received at least one relevant recommendation. The STUFF I'VE SEEN system designed by Dumais et al. (2003) performed a similar function, but was specifically geared towards re-finding documents or Web pages the user had seen before. Certain IMAs focus particularly on acting as a writing assistant that locates relevant references, such as the REMEMBRANCE AGENT by Rhodes and Maes (2000), the PIRA system by Gruzd and Twidale (2006), and the À PROPOS project by Puerta Melguizo et al. (2008). Finally, the well-known CITESEER[5] system originally also offered personalized reference recommendations by tracking the user's interests using both content-based and citation-based features (Bollacker et al., 2000).

Significant work on recommending references for research papers has also been done by McNee (2006), who approached it as a separate recommendation task, and compared different classes of recommendation algorithms both through system-based evaluation and user studies. In McNee et al. (2002), five different CF algorithms were compared on the task of recommending research papers, with the citation graph between papers serving as the matrix of ratings commonly used for CF. Here, the citation lists were taken from each paper and the cited papers were represented as the 'items' for the citing paper. The citing paper itself was represented as the 'user' in the matrix[6]. Citing papers could also be included as items if they are cited themselves. McNee et al. (2002) compared two user-based and item-based filtering algorithms with a Naive Bayes classifier and two graph-based algorithms. The first graph-based algorithm ranked items on the number of co-citations with the citations referenced by a 'user' paper; the other considered all papers two steps away in the citation graph and ranked them on tf·idf-weighted term overlap between the paper titles. They used 10-fold cross-validation to evaluate their algorithms using a rank-based metric and found that user-based and item-based filtering performed best. In a subsequent user study, these algorithms were also the best performers because they generated the most novel and most relevant recommendations. A similar, smaller approach to using the citation graph was done by Strohman et al. (2007), who only performed a system-based evaluation. In later work, McNee et al. (2006) compared user-based filtering, a standard content-based filtering using tf·idf-weighting, and Naive Bayes with a Probabilistic Latent Semantic Analysis algorithm. They defined four different reference recommendation subtasks: (1) filling out reference lists, (2) maintaining awareness of a research field, (3) exploring a research interest, and (4) finding a starting point for research (McNee et al., 2006). They evaluated the performance of their algorithms on these four tasks and found that user-based filtering performed best, with the content-based filtering a close second. In addition, they found that certain algorithms are better suited for different tasks.

[5] http://citeseer.ist.psu.edu/

[6] By using the citation web in this way, and not representing real users as the the users in the ratings matrix they were able to circumvent the cold start problem. This problem is much less pronounced when recommending for social bookmarking websites because users have already implicitly rated many items by adding them to their personal profiles.

Routing and Assigning Paper for Reviewing A task related to recommending references is routing and assigning papers to program committee members for review. Papers are normally assigned manually to reviewers based on expertise area keywords that they entered or knowledge of their expertise of other committee members. Dumais and Nielsen (1992) were among the first to investigate an automatic solution to this problem of paper assignment. They acquired textual representations of the submitted papers in the form of titles and abstracts, and used Latent Semantic Indexing, a dimensionality reduction technique, to match these against representations of the reviewers' expertise as supplied by the reviewers in the form of past paper abstracts. With their work, Dumais and Nielsen (1992) showed it was possible to automate the task acceptably. Later approaches include Yarowsky and Florian (1999), Basu et al. (2001), Ferilli et al. (2006), and Biswas and Hasan (2007). All of them use the sets of papers written by the individual reviewers as content-based expertise evidence for those reviewers to match them to submitted papers using a variety of different algorithms. The most extensive work was done by Yarowsky and Florian (1999), who performed their experiments on the papers submitted to the ACL '99 conference. They compared both content-based and citation-based evidence for allocating reviewers and found that combining both types resulted in the best performance. However, most of the work done in this subfield is characterized by the small size of their data sets; we refer the reader to the references given for more information.

2.1.5 Recommendation in Context

Historically, researchers have focused on building 'more accurate' recommender systems, and have equated this with 'better liked' and 'more useful' without involving the users in this process (McNee, 2006). Indeed, the majority of the related work described so far has focused on experimental validation in a laboratory setting, with only a handful of small-scale user studies. We stated in the previous chapter that we also take a system-based approach to evaluation, and we will describe our reasons in more detail in Subsection 3.4.3. In the current section, we give a short overview of the most important work on how to involve the user in the recommendation process. It is important to establish that user satisfaction is influenced by more than just recommendation accuracy. This was signaled by, among others, Herlocker et al. (2004) and Adomavicius and Tuzhilin (2005). For instance, while a good recommendation algorithm can produce personalized recommendations for each user, the type of personalization applied by an algorithm is exactly the same across all users. This is not beneficial to user satisfaction because not every user request for recommendations is made in the same context. Different contexts can call for different types of personalization and recommendation. Depending on the situation, a user might want to fulfill quite different tasks using a recommender system, and some established algorithms have been shown to be more appropriate for certain tasks than others (McNee et al., 2006). For instance, in research paper recommendation filling out a reference list is rather different from the desire to maintain awareness of a research field, requiring different recommendation algorithms. In addition, the user's interaction with the system and satisfaction with the results depend on a variety of contextual factors such as the user's intentions, his emotional state, and the user' confidence in the system (McNee, 2006).

Context in Information Seeking and Retrieval The observation is also valid in the fields of information seeking and retrieval, where the search process is similarly influenced by the context of the user. The relevance of the same set of returned results for two identical queries can easily differ between search sessions because of this. In the field of information seeking, a number of frameworks for understanding user needs and their context have been developed. Many different theories have been proposed over the years, such as the four stages of information need by Taylor (1968), the theory of sense-making by Dervin (1992), the information foraging theory by Pirolli and Card (1995), and the cognitive theory of information seeking and retrieval by Ingwersen and Järvelin (2005). Common to all of these theories is the importance of understanding the user's information context to increase the relevance of the results (McNee, 2006). In this section, we zoom in on the cognitive theory of information seeking and retrieval (IS&R) by Ingwersen and Järvelin (2005), and describe it in the context of recommender systems. In this theory, the context of an IS&R process is represented as a nested model of seven different contextual layers, as visualized in Figure 2.2.

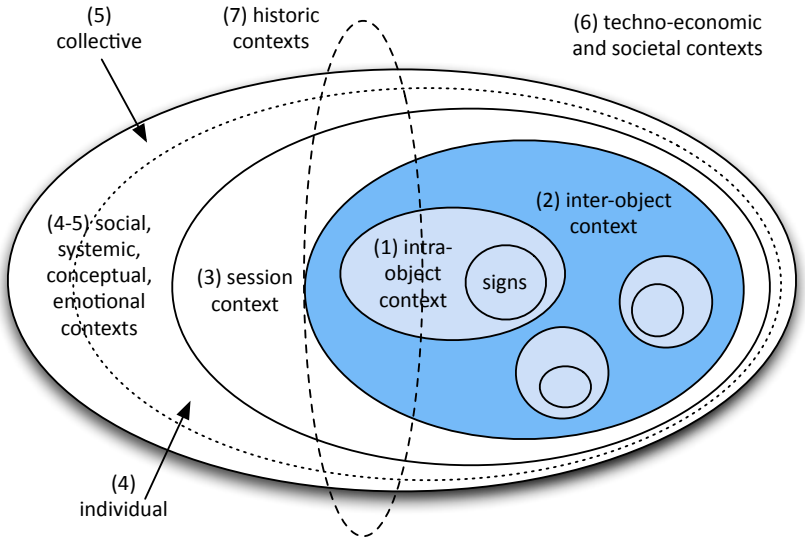


Figure 2.2: A nested model of seven contextual layers for information seeking and retrieval (Ingwersen and Järvelin, 2005). Revised version adopted from Ingwersen (2006).

This model allows us to classify different aspects of the recommendation process into different types of context. All of these seven contextual layers affect users in their interaction with recommender systems. Below we list the seven relevant contextual layers and give practical examples of how they could be quantified for use in experiments.

(1) Intra-object context For recommendation, the intra-object context is the item itself and its intrinsic properties. It can cover a wide range of metadata, depending on the type of item, such as title, author, publication venue, musical genre, director, cast, etc. In case of items with textual content, such as research papers or Web pages, it could also include the structures within the text. However, the structure of multimedia items such as movies or music is more difficult to quantify.

(2) Inter-object context includes the relations between items, such as citations or links between authors and documents in case of research papers. External metadata such as movie keywords, assigned index terms, and tags can also link documents together, as well as playlist structures that group together a set of songs.

(3) Session context involves the user-recommender interaction process and would involve real user tests or simulations of interactions. Observing system usage patterns, such as printing or reading time, would also be context in the case of recommending through IMAs.

(4) Individual social, systemic, conceptual, and emotional contexts If items are linked via a folksonomy, then this could serve as a possible source of social, conceptual, and even emotional context to the different documents. Rating behavior, combined with temporal information can also serve to predict, for instance, emotional context.

(5) Collective social, systemic, conceptual, and emotional contexts An important contextual social aspect of recommending items is finding groups of like-minded users and similar items that have historically shown the same behavior to generate new recommendations. Again, the folksonomy could provide aggregated social, conceptual, and even emotional context to the different documents.

(6) Techno-economic and societal contexts This more global form of context influences all other lower tiers, but is hard to capture and quantify, as it is for IS&R.

(7) Historical contexts Across the other contextual layers there operates a historical context that influences the recommendations. Activity logs of recommender systems would be a appropriate way of capturing such context, possibly allowing for replaying past recommender interaction.

Human-Recommender Interaction Neither the cognitive theory of IS&R nor the other three theories of information seeking we mentioned earlier in this section were originally developed for recommender systems. This means these theories are therefore not fully applicable to the field of recommender systems. McNee (2006) was the first to recognize this lack of a user-centered, cognitive framework for recommendation and proposed a descriptive theory of *Human-Recommender Interaction* (HRI). This singular focus on recommender systems is a major advantage of HRI theory, although it has only been applied and verified experimentally on one occasion.

HRI theory is meant as a common language for all stakeholders involved in the recommendation process—users, designers, store owners, marketeers—to use for describing the important elements of interaction with recommender systems (McNee, 2006). These elements are grouped into three main pillars of HRI: (1) the *recommendation dialogue*, (2) the *recommendation personality*, and (3) the *end user's information seeking tasks*. Each of these three pillars is divided up into so-called *aspects*, which refer to the individual elements of HRI. In its current form as defined by McNee (2006), HRI theory contains a total of 21 aspects. Figure 2.3 shows these 21 of HRI aspects and the three pillars they are grouped under.

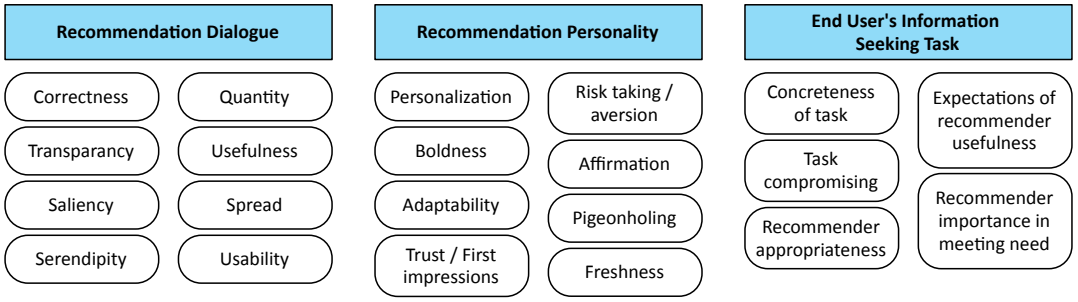


Figure 2.3: A visualization of the theory of Human-Recommender Interaction by McNee (2006). HRI theory consists of 21 interaction aspects, organized into three pillars. Figure taken from McNee (2006).

The aspects can be seen as the 'words' of the shared language that the stakeholders can use to communicate about interaction with recommender system. HRI theory states that each aspect can have both a system-centered and a user-centered perspective. This means that for most aspects it is possible to devise a metric that allows the system designer to measure the objective performance of the system on that interaction aspect. The user's perception of how well the recommender system performs on this aspect does not necessarily have to match the outcome of these metrics however. Both perspectives are seen as equally important in HRI theory.

We will briefly describe each of the three pillars and give a few examples of aspects belong to those pillars. We refer to McNee (2006) for a more detailed description of all aspects and the three pillars. The first pillar, *recommendation dialogue*, deals with the immediate interaction between the user and the recommendation algorithm, which is cyclical in nature. An example aspect here is *transparency*, which means that the user should understand (at a high level) where a recommendation is coming from, for instance, in terms of how an item is similar to items that the user has rated before. Greater transparency has been shown to lead to higher acceptance of a recommender system (Herlocker et al., 2000; Tintarev and Masthoff, 2006).

The second pillar of HRI is the *recommendation personality*, which covers the overall impression that a user constructs of a recommender system over time. Recommender systems are meant to 'get to know the user', which means users can start attributing personality characteristics to the system. An negative example of a personality-related aspect is *pigeonholing*, where a user receives a large number of similar recommendations in a short time, which could change the user's perception for the worse. The item-based CF algorithm, for instance, has shown an aptitude for getting stuck in 'similarity wells' of similar items (Rashid et al., 2002). *Trust* is another important aspect, and is related to the "Don't look stupid" principle formulated by McNee et al. (2006). It states that even a single nonsense recommendation can cause the user to lose confidence in the recommender system, even if the other recommendations are relevant.

The last HRI pillar focuses on the *end user's information seeking task* and the match with the recommendation algorithm. An example is *recommender appropriateness*: not every

recommendation algorithm is suited to each information seeking task, as we argued before. Music recommendation depends largely on the user's personal taste, but more complicated tasks could require more complicated external criteria which might be better served by different algorithms (McNee, 2006).

2.2 Social Tagging

Under the Web 2.0 label, the past decade has seen a proliferation of social websites focusing on facilitating communication, user-centered design, information sharing and collaboration. Examples of successful and popular social websites include wikis, social networking services, blogs, and websites that support content sharing, such as social bookmarking. An important component of many of these services is *social tagging*; giving the users the power to describe and categorize content for their own purposes using *tags*. Tags are keywords that describe characteristics of the object they are applied to, and can be made up of one or more words. Tags are not imposed upon users in a top-down fashion by making users choose only from a pre-determined set of terms; instead, users are free to apply any type and any number of tags to an object, resulting in true bottom-up classification. Users cannot make wrong choices when deciding to apply certain tags, since their main motivation is to tag for their own benefit: making it easier to manage their content and re-find specific items in the future. Many other names have been proposed for social tagging, including collaborative tagging, folk classification, ethno-classification, distributed classification, social classification, open tagging, and free tagging (Mathes, 2004; Hammond et al., 2005). We use the term *social tagging* because it is common in the literature, and because it involves the activity of labeling objects on social websites. We do see a difference between collaborative tagging and social tagging, and explain this in more detail in Subsection 2.2.2. Although there is no inherent grouping or hierarchy in the tags assigned by users, some researchers have classified tags into different categories. A popular classification is that by Golder and Huberman (2006), who divide tags into seven categories based on the function they perform. Table 2.1 lists the seven categories. The first four categories are extrinsic to the tagger and describe the actual item they annotate with significant overlap between individual users. The bottom three categories are user-intrinsic: the information they provide is relative to the user (Golder and Huberman, 2006).

The foundations for social tagging can be said to have been laid by Goldberg et al. (1992) in their TAPESTRY system, which allowed users to annotate documents and e-mail messages. These annotations could range from short keywords to longer textual descriptions, and could be shared among users. The use of social tagging as we know it now was pioneered by Joshua Schachter when he created the social bookmarking site Delicious in September 2003 (Mathes, 2004). Other social websites, such as the photo sharing website Flickr, adopted social tagging soon afterwards[7]. Nowadays, most social content sharing websites support tagging in one way or another. Social tagging has been applied to many different domains, such as bookmarks (`http://www.delicious.com`), photos (`http://www.`

[7] Although Delicious were the first to popularize the use of social tagging to describe content, there are earlier examples of websites that allowed user-generated annotation of content. See Section 2.3 for some examples.

Table 2.1: A tag categorization scheme according tag function, taken directly from Golder and Huberman (2006). The first four categories are extrinsic to the tagger; the bottom three categories are user-intrinsic.

Function	Description
Aboutness	Aboutness tags identify the topic of an item, and often include common and proper nouns. An example could be the tags crisis and bailout for an article about the current economic crisis.
Resource type	This type of tag identifies what kind of object an item is. Examples include recipe, book, and blog.
Ownership	Ownership tags identify who owns or created the item. An example would be the tags golder or huberman for the article by Golder and Huberman (2006).
Refining categories	Golder and Huberman (2006) argue that some tags do not stand alone, but refine existing categories. Examples include years and numbers such as 2009 or 25.
Qualities/characteristics	Certain tags represent characteristics of the bookmarked items, such as funny, inspirational, or boring.
Self-reference	Self-referential tags identify content in terms of its relation to the tagger, such as myown or mycomments.
Task organizing	Items can also be tagged according to tasks they are related to. Popular examples are toread and jobsearch.

flickr.com), videos (http://www.youtube.com), books (http://www.librarything.com), scientific articles (http://www.citeulike.org), movies (http://www.movielens.org), music (http://www.last.fm/), slides (http://www.slideshare.net/), news articles (http://slashdot.org/), museum collections (http://www.steve.museum/), activities (http://www.43things.com), people (http://www.consumating.com), blogs (http://www.technorati.com), and in enterprise settings (Farrell and Lau, 2006).

Early research into social tagging focused on comparing tagging to the traditional methods of cataloguing by library and information science professionals. We discuss these comparisons in Subsection 2.2.1, and describe under what conditions social tagging and other cataloging methods are the best choice. Then, in Subsection 2.2.2, we distinguish two types of social tagging that result from the way social tagging is typically implemented on websites. These choices can have an influence on the network structure of users, items, and tags that emerges, and thereby on recommendation algorithms. We complete the current section in Subsection 2.2.3 by providing some insight into the use of a social graph for representing social tagging.

2.2.1 Indexing vs. Tagging

Early academic work on social tagging focused mostly on the contrast between social tagging and other *subject indexing* schemes, i.e., describing a resource by index terms to indicate what the resource is about. Mathes (2004) distinguishes between three different groups that can be involved in this process: intermediaries, creators, and users. *Intermediary indexing* by professionals has been an integral part of the field of library and information science since its inception. It is aimed at classifying and indexing resources by using thesauri or hierarchical classification systems. By using such controlled vocabularies—sets of

pre-determined, allowed descriptive terms—indexers can first control for ambiguity by selecting which terms are the preferred ones and are appropriate to the context of the intended user, and then link synonyms to their favored variants (cf. Kipp (2006)). According to Lancaster (2003), indexing typically involves two steps: conceptual analysis and translation. The conceptual analysis stage is concerned with determining the topic of a resource and which parts are relevant to the intended users. In the translation phase, the results of the conceptual analysis are then translated into selecting the appropriate index terms, which can be difficult to do consistently, even between professional indexers (Lancaster, 2003; Voß, 2007). Another problem of intermediary indexing is scalability: the explosive growth of content means it is intractable for the relatively small group of professional indexers to describe all content.

Index terms and keywords can also be assigned by the *creators* of a resource. This is common practice in the world of scientific publishing, and popular initiatives, such as the Dublin Core Metadata Initiative[8], have also been used with some success (Mathes, 2004). In general, however, creator indexing has not received much attention (Kipp, 2006). By shifting the annotation burden away from professional indexers to the resource creators themselves, scalability problems can be reduced, but the lack of communication between creators of different resources makes it more difficult to select consistent index terms. A second problem is that the amount and quality of creator-supplied keywords is highly dependent on the domain. For instance, Web pages have been shown to lack useful metadata on many occasions (Hawking and Zobel, 2007) in the context of supporting retrieval. With social tagging the responsibility of describing resources is placed on the shoulders of the *users*. This increases scalability even further over creator indexing, as each user is made responsible for describing his own resources. It also ensures that the keywords assigned to resources by a user are directly relevant to that user.

One of the key differences between the three different indexing schemes is the level of coordination between the people doing the indexing, i.e., what terms are allowed for describing resources. Intermediary indexing requires the highest degree of coordination, whereas social tagging requires no explicit coordination between users. The level of coordination required for creator indexing lies somewhere in between. These differences in coordination were confirmed by Kipp (2006), who investigated the distribution of index terms and tags for the three different indexing methods. For a small collection of scientific articles, Kipp (2006) compared tags, author-assigned keywords, and index terms assigned by professional indexers. She showed that the distribution of terms of the latter two indexing approaches was different from the tag distribution, which showed a considerably longer tail of tags that were assigned only once. The larger variety of tags is a direct results of the lower level of coordination. Her findings hinted at the presence of a power law in the tag distribution, which was later confirmed by, among others, Shen and Wu (2005) and Catutto et al. (2007).

It is important to remark that, despite its growing popularity, social tagging is not necessarily the most appropriate method of describing resources in every situation. In certain situations, indexing by intermediaries is still the preferred approach according to Shirky (2005). Such situations are characterized by a small, stable collection of objects with clear,

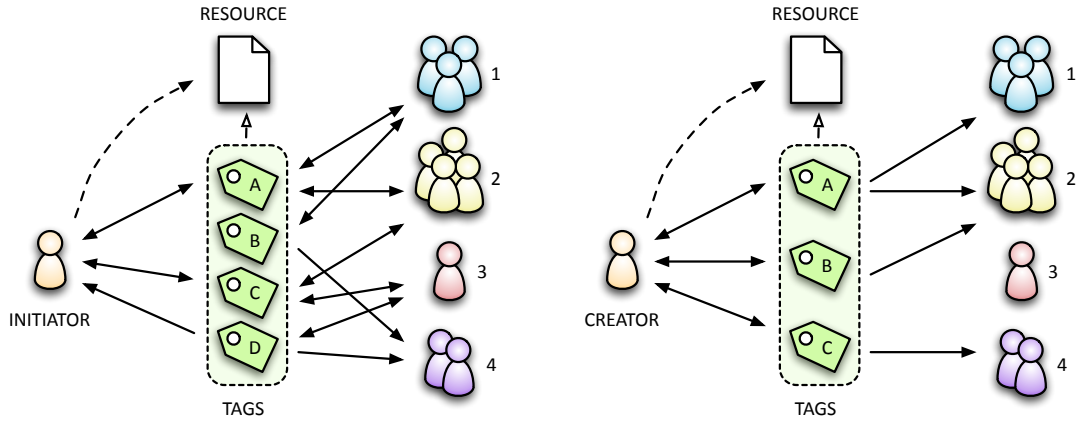
[8] http://dublincore.org/

formal categories. The cataloguers themselves have to be experts on the subject matter, but users have to be experts in using the classification system as well. Examples of such collections include library collections or the periodic table of elements. In contrast, social tagging works best for large, dynamic, heterogeneous corpora where users cannot be expected to gain expertise in a coordinated classification scheme (Shirky, 2005). The World Wide Web is a good example of such a scenario with billions of Web pages that vary wildly in topic and quality.

2.2.2 Broad vs. Narrow Folksonomies

The aggregation of the tagging efforts of all users is commonly referred to as a *folksonomy*, a portmanteau of 'folk' and 'taxonomy', implying a classification scheme made by a group of users. Like the hierarchical classification schemes designed by professional indexers to organize knowledge, a folksonomy allows any user to navigate freely between items, tags, and other users. The term was coined by Vander Wal (2005b), who defines a folksonomy as the result of "personal free tagging of information and objects (anything with a URL) for one's own retrieval". Different variations on this definition have been proposed in the past. In some definitions, only tags that are assigned by a user for his own benefit as considered to be a part of the folksonomy. Consequently, tagging for the benefit of others is excluded from the folksonomy according to this definition (Lawley, 2006). We refer the reader to Chopin (2007) for an extensive overview and discussion of the different definitions. For the recommendation experiments described in this thesis, we do not take into account the different motivations that users might have when tagging items. We define a folksonomy as "an aggregated network of users, items, and tags from a single system supporting social tagging". Vander Wal (2005a) distinguishes between two types of folksonomies, depending on how social tagging is implemented on the website: *broad folksonomies* and *narrow folksonomies*. Figure 2.4 illustrates these two types of folksonomies.

The essential difference between a broad and a narrow folksonomy is who is allowed to tag a resource: every user interested in the resource, or only the creator of the resource. This dichotomy is caused by the nature of the resources being tagged in the system. *Broad folksonomies* emerge in social tagging scenarios where the resources being tagged are publicly available, and were not necessarily created by the people who tagged and added them. A good example are Web page bookmarks: any user can bookmark a Web page and many pages will be useful to more than one user. More often than not, the bookmarked Web pages were not created by the user who added them to his profile. This means that interested users will add their own version of the bookmarked URL with their own metadata and tags. Figure 2.4(a) illustrates this case for a single example resource. The 'initiator', the first user to add the resource to the system, has tagged the resource with tags A and C when he added it to the system. Users in group 1 added the post with tags A and B, group 2 with tags A and C, and group 3 with tags C and D. Notice that tags B and D were not added by the original creator, although it is possible that the initiator later retrieves the resource with tag D. The two users in group 4 never add the resource, but find it using tags B and D later. Figure 2.4(a) illustrates the collaborative nature of tagging the resource: a tag can



(a) Broad folksonomy (b) Narrow folksonomy

Figure 2.4: Two types of folksonomies: broad and narrow. The figure is slightly adapted from Vander Wal (2005a). The four user groups on the right side of each figure denote groups of users that share the same vocabulary. An arrow pointing from a user (group) to a tag means that the tag was added by that user (group). An arrow pointing from a tag to a user (group) means that the tag is part of the vocabulary of that user (group) for retrieving the resource. The creator/initiator is the user who was the first to create or add the resource to the system.

be applied multiple times to the same resource. This is why we refer to this scenario as *collaborative tagging*.

In contrast, a *narrow folksonomy*, as illustrated in Figure 2.4(b), emerges when only the creator of a resource can tag the item. For example, in the case of the video sharing website YouTube, a user can create a video and upload it to the website. This user is the original creator of the video and adds tags A, B, and C to the video to describe it after uploading. As a consequence, each tag is applied only once to a specific resource in a narrow folksonomy. We therefore refer to this scenario as *individual tagging*. All other users are dependent on the creator's vocabulary in a narrow folksonomy: users in group 1 can use tag A to locate the resource, users in group 2 may use tags A and B, and group 4 users may use tag C. User 3, however, cannot find the resource because his vocabulary does not overlap with that of the creator. In this thesis we look at recommendation of bookmarks and scientific articles. These items are typically added and tagged by many different users, and result in broad folksonomies. We do not focus on recommendation for individual tagging systems, only for collaborative tagging.

2.2.3 The Social Graph

Figure 2.4 illustrates the concept of a folksonomy for a single item, but this mini-scenario occurs for many different items on websites that support social tagging. With its myriad of connections between users, items, and tags, a folksonomy is most commonly represented as a undirected tripartite graph. Mika (2005) and Lambiotte and Ausloos (2006) were the first to do so. Figure 2.5 illustrates such a *social graph*.

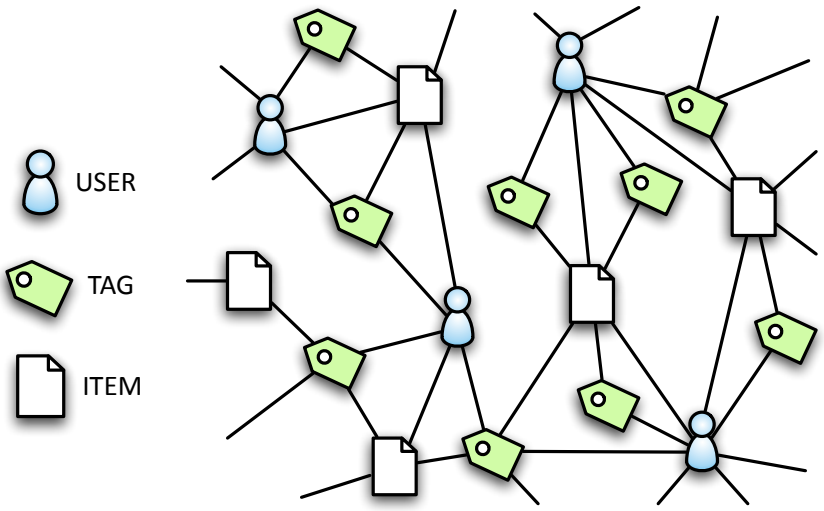


Figure 2.5: Visualization of the social graph as an undirected tripartite graph of users, items, and tags.

We take the tripartite graph as the basis for our experiments in Chapter 4. It is important to note, however, that the three different types of nodes in the graph—users, items, and tags—do not all have the same roles. Users are active participants in the graph: they add items and label them with tags. Items and tags are passive nodes and can be seen as content bearers. A fitting metaphor here is that of the user as a predator, snaring items as his prey using tags. This implies that is possible to represent the folksonomy in a different type of graph representation with directed edges, although we do not consider this in the thesis due to temporal restrictions.

2.3 Social Bookmarking

As mentioned earlier, the Web 2.0 phenomenon caused a shift in information access and creation from local and solitary, to global and collaborative. *Social bookmarking* websites are a prime example of this: instead of keeping a local copy of pointers to favorite Web pages, users can instead store and access their bookmarks online through a Web interface on a remote Web server, accessible from anywhere. The underlying application then makes all stored information shareable among users. In addition to this functionality, most social bookmarking services also enable social tagging for their users, in addition to standard metadata such as the title and a brief description of the bookmarked Web page.

The current generation of social bookmarking websites is not the first. One of the original online bookmarking services was the now defunct itList, which was originally created in April 1996 (De Nie, 1999). It allowed users to store and organize their bookmarks online and, if they chose to do so, share them with other users. itList also enabled users to sort their bookmarks using a variety of categories, although it did not allow for free categorization. In the years that followed, several competing services were launched, such as Backflip, Blink, ClickMarks, and HotLink, each attempting to offer unique features to attract users (Lawlor,

2000). For instance, ClickMarks offered automatic categorization to sort bookmarks into folders. None of these services survived the bursting of the 'dot-com' bubble. The second wave of social bookmarking services started in September 2003 with the creation of Delicious by Joshua Schachter (Mathes, 2004). The instant popularity of Delicious led to the launch of many other Web 2.0-style social bookmarking services, such as Diigo, Simpy, Ma.gnolia and Mister Wong[9]. The main differences between the two waves of social bookmarking services are a stronger emphasis on sharing one's bookmarks with other users. and the addition of social tagging, as pioneered by Delicious. Some social bookmarking services, however, only allow users to share bookmarks without any support for social tagging, such as the GiveALink service (Stoilova et al., 2005) and StumbleUpon[10].

Below we describe domains fit for social bookmarking (Subsection 2.3.1), and how users commonly interact with a social bookmarking system (Subsection 2.3.2). We conclude the section and this chapter in Subsection 2.3.3 by giving an overview of the three research tasks that have received the majority of the attention in the related work.

2.3.1 Domains

So far we have defined social bookmarking services as websites that focus on the bookmarking of Web pages. In addition to Web bookmarks, there are three other domains for social 'storage' services that are worth mentioning. We already mentioned that some social bookmarking websites operate in the domain of scientific articles and research papers. These services are known as *social reference managers* or *social citation managers*. They allow users to store, organize, describe, and manage their collection of scientific references. Although more general-purpose websites like Delicious could conceivably be used for this as well, social reference managers offer specialized features, such as article-specific metadata, creating reading lists, reading priorities, and export facilities for different formats such as BibTeX, RIS, and EndNote. Examples of social reference managers include CiteUlike (http://www.citeulike.org/), Connotea (http://www.connotea.org/), BibSonomy (http://www.bibsonomy.org), and refbase (http://www.refbase.org/), the first three of which all support collaborative tagging.

Next to the domains mentioned above, a fourth third domain for social information storage and management is books. *Social cataloging services* allow users to store, describe, and manage books that they own or have read. Typically, social cataloging services allow users to rate books, and to tag and review them for their own benefit and the benefit of other users. They typically also use identifiers such as ISBN or ISSN numbers to automatically retrieve the book metadata from centralized repositories such as the Library of Congress or Amazon. Examples include LibraryThing (http://www.librarything.com/), Shelfari (http://www.shelfari.com/), aNobii (http://www.anobii.com/), GoodReads (http://www.goodreads.com/), and WeRead (http://weread.com/). The first three support collaborative tagging.

[9] We refer the reader to http://en.wikipedia.org/wiki/List_of_social_bookmarking_websites for an up-to-date list of social bookmarking services.

[10] http://www.stumbleupon.com/

The fourth example of collaborative information sharing services are formed by the so-called *social news websites*, that allow users to share and discover any kind of online content, but with an emphasis on online news articles. After a link or news story has been submitted, all website users get to vote and comment on the submissions, and stories are ranked on these reactions. Only stories with the most votes appear on the front page. The most popular social news websites are Digg (http://www.digg.com/), Reddit (http://www.reddit.com/), Fark (http://www.fark.com/), and Mixx (http://www.mixx.com/).

2.3.2 Interacting with Social Bookmarking Websites

How can users typically interact with a social bookmarking website? Because of the public, shared nature of Web pages, social bookmarking websites allow for collaborative tagging. The broad folksonomy that emerges from this results in a rich network of connections between users, items, and tags, which is reflected in the navigational structure. Figure 2.6 shows the typical navigating and browsing structure of a social bookmarking website; we take the popular service Delicious as our example.

The personal profile page is the starting point for every user (top left in Figure 2.6). It lists the assigned tags and metadata for every bookmark that a user has added. In addition, each item and tag on the page is linked to a separate page that chronologically show the activity of those tags and items. For a selected tag, clicking on it leads the user to a page that shows all other items in the system that have been annotated with that tag (bottom left). The popularity of the bookmarked item on Delicious is shown for each post. When a user clicks on this, they are forwarded to the item's history page (bottom right), which show all other users who have added the selected item. For each post of that item, it also lists what tags and metadata were assigned to it by which other user. Another typical feature of social bookmarking—and websites supporting social tagging in general—is a visual representation of all of a user's tags, known as a *tag cloud* (top right). In a tag cloud, tags are sorted alphabetically and the popularity of a tag (or rather usage intensity) is denoted by varying the markup of the link: larger font sizes and/or darker font colors tend to denote more popular tags.

Some social bookmarking websites offer extra features on top of this: Diigo[11], for instance, also allows its users to highlight the Web pages they bookmark and add notes to them, which are stored in the system and overlaid on the bookmarked Web pages when the user revisits them. Faves[12] is unique in being the only social bookmarking website to allow its users to explicitly rate the items they have added on a five-point scale. Social cataloging applications like LibraryThing and Shelfari also allow explicit ratings. It is interesting to note, however, that some users find a way around the absence of such features. On Delicious, for example, tags such as *, ***, or ***** are used by several people to represent the quality of a bookmarked Web page.

[11] http://www.diigo.com/
[12] http://www.faves.com/

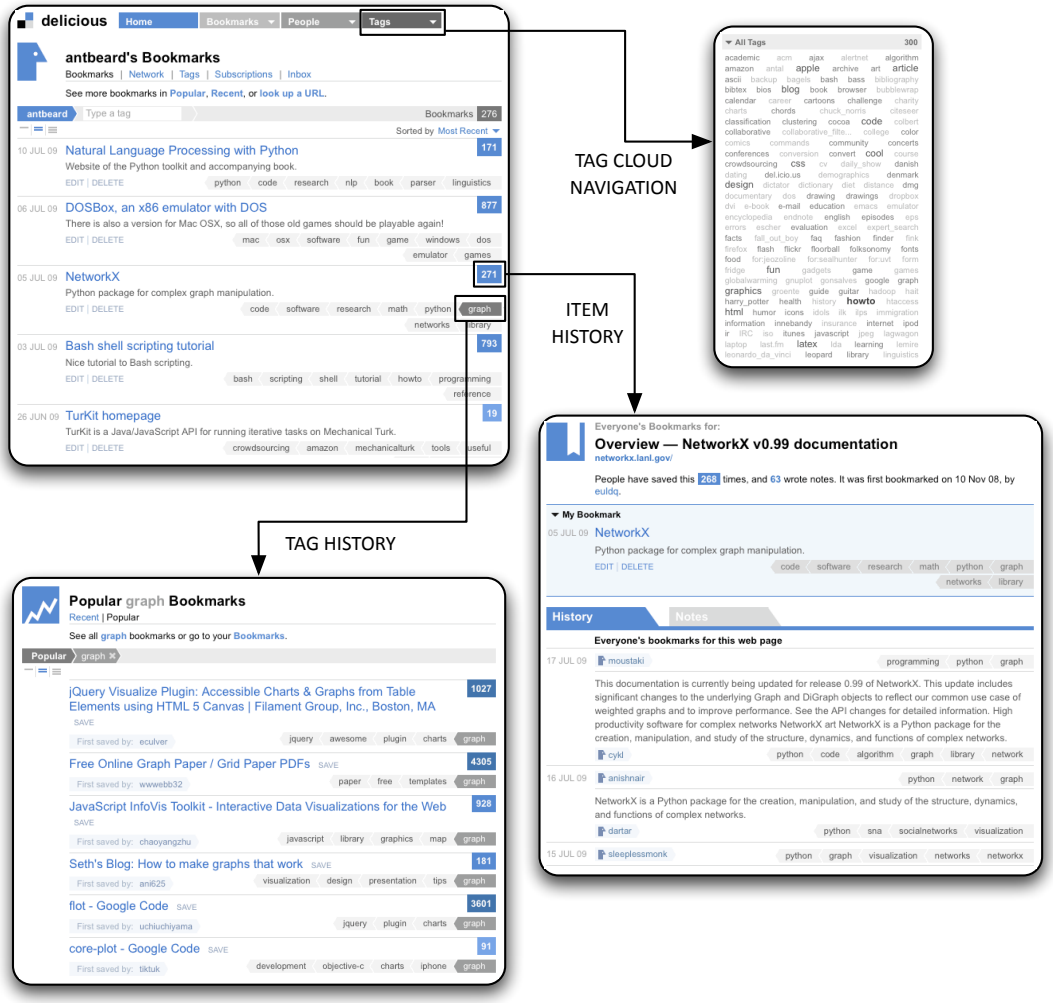


Figure 2.6: Navigation on a social bookmarking website. The starting point for every user is their profile page which lists the bookmarks they added to their profile (top left). From there, users can browse to tag pages (bottom left) which show which other bookmarks have been tagged with the selected tag; and to item history pages (bottom right), which show all other users who have added the selected item, and what tags and metadata they assigned to it. Users can also get an overview of the tags they have used from their tag cloud view (top right), which marks up the more popular tags with a darker font color and a larger font size.

2.3.3 Research tasks

We mentioned earlier that there is not a large body of related work on the task of item recommendation on social bookmarking websites. In this section we briefly discuss the tasks and problems that *have* seen more attention: browsing, search, and tag recommendation[13].

Browsing One of the strengths of social tagging as a way of categorizing and describing resources is that users are free to tag without any considerations of relationships between

[13]These are not the only possible tasks that can be supported on social bookmarking websites; we refer the reader to Clements (2007) for an insightful overview.

tags. However, it is obvious that certain tags will be more related to each other than others, because they are synonyms, hyponyms, or because there is another kind of conceptual relationship between them. Several researchers have proposed techniques that could improve the browsing experience. One example is automatically deducing a tag hierarchy for improved browsing as proposed by Heymann and Garcia-Molina (2006) and Li et al. (2007). A second way of improving the browsing experience is by clustering related bookmarks together, as proposed by Capocci and Caldarelli (2007) and Clements et al. (2008b). Tag clouds do not take such implicit relationships into account and just visualize tags in an alphabetical list, highlighted according to popularity. Garama and De Man (2008) performed an extensive user study to investigate the usefulness of tag clouds for image search. They compared tag clouds as a browsing mechanism to a standard search engine for a known-item retrieval task, and found that, although performance as measured by success rate is higher when using a search engine, users are more satisfied when browsing by using the tag cloud than when using a search engine. This shows the value of a simple representation such as the tag cloud.

Search A second popular research problem in social bookmarking is whether tags can be used to improve (1) retrieval of items on a social bookmarking website and (2) Web search in general. Since tags are condensed descriptions of content, it is possible that they can replace or augment standard retrieval algorithms; this possibility was already hypothesized early on by, for instance, Mathes (2004). Heymann et al. (2008a) tried to answer the question *Can social bookmarking help Web search?* by analyzing various properties of Delicious[14]. Heymann et al. (2008a) found that Web pages posted to Delicious tend to be quite dynamic and updated frequently. An analysis of the tags showed that they were overwhelmingly relevant and objective. However, compared to the Web, Delicious still only produces small amounts of data, limiting its possible impact on improving Web search. The tags themselves were found to occur in over half of the pages they annotate, and in 80% of the cases they occurred in either the page text, the bookmark title or the Web page URL. Heymann et al. (2008a) therefore concluded that it is unlikely that tags will be more useful than a full text search emphasizing page titles, and that Web search engine performance is therefore unlikely to be impacted. Bischoff et al. (2008) repeated the work of Heymann et al. (2008a) for two more social bookmarking systems, Flickr and Last.fm, and were more positive about the potential for tags. They report a higher percentage of uniqueness of tags compared to anchor and page text, especially in the music domain, where content analysis is more difficult. They conclude that for certain domains tags can have a large impact on search engine performance. Morrison (2007) directly compared the performance of social bookmarking websites with Web search engines and Web directories. He found that there was only a small overlap in the results of each type of retrieval 'tool', with 90% of the results being unique to the system that retrieved them. The largest of the social bookmarking websites performed significantly better than the Web directories. Social bookmarking websites were also better than search engines on dynamic queries such as current events, which confirms the findings of Heymann et al. (2008a).

[14] A year earlier, Yanbe et al. (2007) performed the same kind of analysis on Japanese social bookmarking websites and came to similar conclusions.

Several approaches have been proposed for how tags can be incorporated into retrieval algorithms. Bao et al. (2007) proposed an algorithm based on PageRank (Page et al., 1998) called *SocialPageRank* to measure the static ranking of a Web page based on the tags assigned to it instead of the hyperlinks between pages. They combined it with a simple vector space model to perform Web page retrieval and showed that re-ranking the search results yielded significant improvements over their document retrieval baseline. Zhou et al. (2008) proposed a unified framework that combines the modeling of tag usage with a language modeling approach to IR by enhancing the document and query language models with the tags. Their experiments on a sample of the Delicious website showed significant improvements over a traditional retrieval algorithm. Similar experiments were performed by Carman et al. (2008), who used a similar algorithm to construct probabilistic user profiles to combine with regular document retrieval. In addition, they compared creating (1) user profiles based on tags and (2) profiles based on the content of the bookmarked pages, and showed that a combination performed best.

Tag recommendation The most popular recommendation task in the literature on social bookmarking so far has been tag recommendation: suggesting appropriate tags to a user when he posts a new bookmark to the system. For instance, the BibSonomy system includes a tag recommender that offers up to ten recommended tags when a user finds an interesting Web page or publication and posts it to BibSonomy. The experiments behind this tag recommendation are described in Jäschke et al. (2007b). They found that a graph-based approach to tag recommendation significantly outperformed collaborative filtering approaches. Collaborative filtering was first applied to the problem of recommending tags in 2006 by Xu et al. (2006). Sigurbjörnsson and Van Zwol (2008) aggregated tag occurrence information in different ways to recommend tags for photos on Flickr. Heymann et al. (2008b) experimented with tag prediction by mining tag-based association rules on Delicious using their Stanford Tag Crawl data set from Heymann et al. (2008a). Song et al. (2008) used a combination of graph partitioning and a two-way Poisson mixture model to cluster tags into separate clusters with good results on their two data sets based on Delicious and CiteULike. Chirita et al. (2007) proposed a tag recommender that mined the bookmarked Web pages and pages similar to it by ranking the terms in those documents by their tf·idf value and recommending the top N terms. Blogs are an additional area where tag prediction has been applied to blog posts by Mishne and de Rijke (2006). For each untagged blog post, they first locate similar blog posts using a retrieval system and then assign the most popular tags associated with those best-matching posts to the untagged post (Mishne, 2006).

As a final note, it is worthwhile to ask the question whether these tag recommendations are actually accepted by the user, and what the effects are on the system? Do people converge to a single vocabulary by accepting the suggestions of the tag recommender, or are they stubborn and do they stick to their own choice of tags? Suchanek et al. (2008) examined the influence that these tag recommendations can have on a social bookmarking system and its folksonomy. They found that up to one in three tags are applied purely because of the presence of the tag suggestions. This implies that the distribution of tag popularity in a folksonomy might become biased as a result of offering tag suggestions. However, tag suggestions do provide a form of control over the tag vocabulary of the users.

I

Recommending Bookmarks

In Chapter 1, we described two important characteristics of social bookmarking systems. The first was the presence of the folksonomy, a collaboratively generated categorization of items in a system by its users. This extra annotation layer of collaboratively generated tags binds the users and items of a system together. The second observation about social bookmarking systems is that, in addition to tags, they contain a large amount of additional metadata describing the content of those systems and its users. In the first part of this thesis, we will investigate how to exploit these two characteristics, both separately and combined, to provide the best possible recommendations for users of social bookmarking systems. This leads us to our first three research questions.

RQ 1 How can we use the information represented by the folksonomy to support and improve the recommendation performance?

RQ 2 How can we use the item metadata available in social bookmarking systems to provide accurate recommendations to users?

RQ 3 Can we improve performance by combining the recommendations generated by different algorithms?

Part I is organized as follows. We begin in Chapter 3, where we present the building blocks of our quantitative experimental evaluation. We formally define our recommendation task, and introduce our data sets. We discuss the filtering we applied to the data, our experimental setup, and discuss the evaluation metrics we use. We also critically discuss the pros and cons of our methodological decisions. In Chapter 4, we propose and compare different options for using the folksonomy for item recommendation on social bookmarking websites (RQ 1). Chapter 5 examines how we can use the metadata present in social bookmarking systems to improve recommendation performance (RQ 2). Chapter 6 concludes Part I and investigates if we can improve upon our best-performing algorithms from Chapters 4 and 5 by combining the output of the different algorithms into a new list of recommendations (RQ 3).

Chapter 3

Building Blocks for the Experiments

In Chapter 1 we introduced two features of social bookmarking systems that we believe have the potential to improve recommendation: (1) the folksonomy, serving as an extra layer of information that binds users and items together, and (2) the presence of metadata. In order to evaluate the usefulness of different recommender algorithms for social bookmarking services and to determine how best to incorporate these extra sources of information, we need a proper framework for experimentation and evaluation. While we recognize that recommendation accuracy alone does not guarantee users a satisfying experience, we focus on a quantitative, system-based evaluation of recommender system performance in this thesis. It is possible to simulate, to a satisfying degree, the reaction of the user to different variants of the recommender algorithms and thus determine which algorithms are better suited to which domains (Herlocker et al., 2004). This can be done automatically without any user intervention.

In this chapter we will describe the building blocks for our experiments and their evaluation. First, we need to understand the type of user task we are trying to fulfill with our recommender system, as the rest of the experimental evaluation depends on the choice of recommendation task. Section 3.1 gives an overview of six possible item recommendation tasks and explains which recommendation task we focus on in this thesis. Section 3.2 then describes the four data sets we collected from BibSonomy, CiteULike, and Delicious to evaluate our recommendation algorithms. Section 3.3 describes how we represent our data. We conclude this chapter by describing our experimental setup and evaluation for our specific item recommendation task in Section 3.4.

3.1 Recommender Tasks

A prerequisite of the entire experimental evaluation process is deciding which recommender task we want to support. The choice of data sets, experimental setup, and evaluation metrics all depend on which specific recommender task we select. Recommender systems can fulfill different tasks for different stakeholders (Herlocker et al., 2004). First and foremost

among these stakeholders are the end users, who should be provided with novel and interesting recommended items. However, recommender systems can also be designed, tweaked, or tuned to serve the goals of other system stakeholders. Marketers may want the recommender system to promote new albums or artists for a certain period of time, while other stakeholders such as store owners may wish to recommend excess inventory items. Herlocker et al. (2004) identify six different user tasks for which recommender systems can be used. In this section we describe these six tasks and explain which one is applicable for our scenario of recommendation for social bookmarking.

Annotation in Context In the *Annotation in Context* task, the recommender system generates its suggestions in context, helping the user to make decisions. An example would be an IMA that aids a user during the writing process, recommending relevant literature depending on the user's task and the content of the document being written (Puerta Melguizo et al., 2008). We do not focus on this user task in this thesis.

Find Good Items In many systems users are provided with a ranked list of recommended items, often without the predicted scores or ratings, such as in Web search engines. In such a situation it might be best for a system to generate a few highly useful recommendations even at the cost of being wrong about the others. Presenting the user with a list of top items[1]—the 'best guesses' made by the system—is the most useful scenario we envision for recommendation for social bookmarking services, and the one we will address in this thesis. We will try to recommend the best, novel items for a user based on the bookmarks or scientific articles that user has added to his profile in the past.

Find All Good Items In special cases of the *Find Good Items* task, coverage can be essential, such as in legal search engines or patent search. Here it is important to optimize on recall and find all the good items in the collection. Although it depends on the situation and the type of social bookmarking service, we do not believe that finding all good items is essential for our recommendation experiments.

Recommend Sequence In this task the goal is to predict a useful or interesting sequence of recommended items over time. Ordering is of much greater importance for this user task. Examples include generating smooth music playlists that prevent the so-called iPod whiplash phenomenon, where the listener is confronted with clashing transitions from one musical genre to a completely unrelated one. A second example would be creating reading lists meant to familiarize researchers with a certain field ("First read this introduction, then that survey, ..."). In this thesis we do not focus on this recommendation task.

Just Browsing Sometimes it is rewarding for users to browse around a website and explore the recommendations without an ulterior motive. On social bookmarking websites, the folksonomy connecting users and items through tags can enhance the user's browsing experience greatly and increase the possibility of serendipitous findings. In these browsing scenarios, the quality of the user interface is usually deemed more important than recommendation accuracy. This task is related to the recommendation

[1] Also known as Top-N recommendation.

dialogue aspects of human-recommender interaction we discussed in Section 2.1.5. We do not focus on this user task in this thesis.

Find Credible Recommender In this user task, users tend to play around with the recommender system specifically to see if it will correctly learn their personal preferences and generate trustworthy and credible recommendations. This task is related to the recommendation personality aspects of human-recommender interaction we discussed in Section 2.1.5. This user task is not applicable in our situation either.

3.2 Data Sets

In the previous section we selected the *Find Good Items* task as the item recommendation task we wish to address in this thesis. The next step in our quantitative evaluation is collecting data sets that we can use to evaluate the recommendation algorithms we will design to address this recommender task. In this section we discuss three important issues for collecting such realistic data sets. We look at the field of IR, and adapt three common guidelines for constructing IR data sets for our own purposes. We describe the four data sets we have collected and list statistics of the raw, unfiltered versions of these data sets. Then, in Subsections 3.2.1 through 3.2.3 we discuss the specifics of how these data sets were collected and of the social bookmarking websites they originate from.

In order to test our algorithms, we need to use data sets that are realistic representations of the daily operation of a social bookmarking website. There are three important issues that surface when deciding how to evaluate recommendation in a social bookmarking situation.

Difficult to generalize Using a data set collected from one particular social bookmarking service does not necessarily mean that any findings are generalizable to all social bookmarking services available on the Web. Certain domains might very well have different demands and characteristics that make it hard or even impossible to apply the findings of our experiments there. To counter this, one should always work with a reasonable abstraction of a realistic task and as many different, yet realistic data sets as possible. This way we can obtain an estimate of how well results and conclusions carry over to other social bookmarking settings.

Personal judgments and interestingness *Relevance* in IR is not quite the same as the notions of relevance or *interestingness* in recommendation. Whereas personal judgments from the user can be substituted by judgments from impartial, third-party judges to a certain extent in the field of IR, there is no evidence that this is the same for recommender systems.

Private vs. public data Many social bookmarking websites, such as Delicious and CiteULike, allow its users to keep part or all of their bookmarked items private. Harvesting a data set from the public-facing side of a social bookmarking website therefore means that not necessarily all data is available for testing.

To address the first issue of generalizability, we introduce and use four different data sets from two different domains: scientific articles and Web bookmarks. This will enable to compare our algorithms under different conditions and determine if our findings carry over to other websites and domains. Our four different data sets were taken from three different social bookmarking websites: BibSonomy, CiteULike, and Delicious. One of the websites, BibSonomy, supplies two data sets as it allows its users to post both scientific articles and Web bookmarks to their profiles. We therefore split the BibSonomy data into a data set containing only scientific articles and a data set containing only Web bookmarks. In the remainder of this thesis, we will refer to the BibSonomy data sets containing scientific articles and Web bookmarks as BibArt and BibBoo respectively to avoid confusion. The website CiteULike also covers scientific articles while Delicious covers Web bookmarks. This means we have two data sets corresponding to the domain of scientific articles (BibSonomy and CiteULike), and two covering the domain of Web bookmarks (BibSonomy and CiteULike). With two pairs of data sets sharing the same domain, we can directly examine and compare if the findings from one data set are generalizable to other social bookmarking websites in the same domain or in a different domain altogether.

With regard to the second issue of the difference between relevance and interestingness, the fundamental differences and similarities between these two notions are not covered in this thesis. We therefore abstract away from this and treat relevance the same as interestingness. Extensive user evaluations, such as those performed by McNee (2006) could shed more light on this difference. However, this is beyond the scope of this thesis.

We do not believe the third issue to have a large influence on our experimental setup and results. Although we have no way of knowing how substantial a part of the social bookmarking website's database is missing from collecting a crawl of the public-facing section, we do not believe this data to be radically different in either structure or topic. Moreover, public bookmarks are likely to comprise links to personal log-in pages that would not make for suitable recommendations to other users at any rate. This means we assume any conclusions drawn from our experiments on the public-facing sides of social bookmarking websites to carry over to the private part without any problems.

Constructing Proper Data Sets for Recommendation As far as we know, Herlocker et al. (2004) are the only ones to have explicitly discussed the challenges of creating and using data sets specifically for testing recommender algorithms. They stress that it is "very important that the tasks your algorithm is designed to support are similar to the tasks supported by the system from which the data was collected" (Herlocker et al., 2004, p. 15). An example would be the MovieLens[2] movie recommender system, which offers support for the *Find Good Items* user task. This means that the recommender system tends to show those items it is most confident about, subsequently resulting in more ratings for these good and often more popular movies. As a consequence of this "rich get richer" effect, the MovieLens data set is less likely to have many ratings for the more unpopular movies. Using this data set to evaluate a different task, such as *Recommend Sequence* or *Annotation in Context* would therefore be inappropriate (Herlocker et al., 2004). Hardly any social bookmarking services offer item recommendation to their users at this point and certainly no data sets based on

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

these recommender systems have been made available yet[3], so this could not be taken into account when gathering our data sets.

IR and recommendation are considerably similar in many aspects, such as the construction of test collections for IR and the construction of good data sets for recommendation. According to Croft et al. (2009), there are three important guidelines when creating test collections for IR experiments. We have adapted them to recommendation purposes and list them here.

Guideline 1 It is important to use a collection that is representative of the application in terms of the number, size, and type of resources. This echoes the point made when describing the first issue in the beginning of this section: a good data set is a realistic representation of actual content present in a social bookmarking service.

Guideline 2 Queries should be representative of the real-world application. In our recommendation scenario, queries correspond to the user profiles present on the social bookmarking website—a single user profile is matched against the entire database just as single queries are matched against the document collection. We therefore recommend collecting a realistic and sufficiently large set of user profiles in order to create a good recommendation data set.

Guideline 3 Relevance judgments should be created by people who posed the queries or independent judges. This guideline is especially strict in the case of recommender data sets: judging whether an item is interesting or not is an strictly personal matter, and cannot be done by third parties on behalf of the user for whom the recommendation was made.

With regard to the third guideline, it is necessary to remark that it is impossible to have a *complete* set of user–item judgments. Having each user judge all the items on a social bookmarking website is simply not achievable. This is similar to IR evaluation, where it is practically impossible to judge all query-document pairs for topical relevance. However, according to Carterette et al. (2006), it is better for the quality of the evaluation to judge more queries than to judge more documents. If a small number of queries are used, the results should only be considered indicative rather than conclusive. Since all of our data sets contain thousands of users profiles that serve as our queries, we hope that—analogous to the findings of Carterette et al.—having many different users profiles instead of more judged user–item pairs mitigates the issue of lacking human judgments to a certain extent.

Collecting the Data Sets The field of IR has benefited greatly from the large-scale evaluation initiatives such as the Text REtrieval Conference (TREC)[4]. TREC is comprised of different research tracks, with each track focusing on a specific sub-problem of IR, such as Web search, blog search, or entity retrieval. Each tracks works with shared data sets of realistic size that enable the participants to compare their approaches with each other.

[3] Of course it is entirely possible that a user added bookmarks to, say, his Delicious account because he or she saw them on the *Recent* or *Popular* lists, but we have no way of knowing why a user added a specific bookmark.

[4] Available at `http://trec.nist.gov/`.

Unlike these large-scale initiatives for IR, however, there were no publicly available data sets of social bookmarking websites when the work described in this thesis was started. We therefore had to collect the CiteULike and Delicious data sets ourselves. In addition, the BibSonomy collection was added when it became available in May 2008 as part of the 2008 ECML/PKDD Discovery Challenge. We understand that the quality of any scientific experiment is dependent on the possibility of verification and comparison of the results. We therefore make these data sets publicly available to the recommender systems community[5]. For each data set, we removed the spam content from our crawls wherever possible, as spam users and content will only serve to disturb our recommendation experiments.

Table 3.1 lists some of the basic statistics of the raw, unfiltered data sets we collected. For each of the four data sets, we list the size in terms of users, items, tags, and posts. In addition, we show the average and maximum counts of the relationships of one object type to another. For instance, the average number of tags per user for CiteULike is equal to 27.2, while the maximum numbers of users that share an item in Delicious is 764 in our raw Delicious data set. We do not list minimum counts for any of the calculated measures, because they are always equal to 0 or 1, depending on the measure. We also list information about the sparsity of each of the data sets.

Table 3.1: Statistics of the four different data sets used as the basis for our experiments.

	bookmarks		**articles**	
	BibSonomy	**Delicious**	**BibSonomy**	**CiteULike**
# users	1,913	17,975	1,310	18,072
# items	157,304	3,870,535	136,890	585,351
# tags	44,194	554,856	27,702	129,409
# posts	177,334	5,975,514	156,124	672,396
avg. # items per user	92.7	332.4	119.2	37.2
avg. # users per item	1.1	1.5	1.1	1.1
avg. # tags per user	59.1	238.7	43.9	27.2
avg. # users per tag	2.5	7.1	2.1	3.3
avg. # tags per item	3.6	3.7	1.9	3.3
avg. # items per tag	12.6	24.6	9.5	13.7
max # items per user	33,381	15,044	57,739	6,581
max # users per item	54	764	45	546
max # tags per user	11,165	6,771	4,767	4,045
max # users per tag	257	8,974	490	1,399
max # tags per item	101	548	143	1,282
max # items per tag	35,561	121,274	65,923	150,979
% users without tags	0	7.57	0	14.22
% items without tags	0	6.01	0.16	8.40
% posts without tags	0	6.00	0.19	9.23
user-item sparsity (%)	99.9411	99.9914	99.9129	99.9936

The origins of our four data sets are described in greater detail in Subsections 3.2.1 through 3.2.3. We briefly describe the spam problem for each of the data sets in those sections. In Chapter 7 we examine the presence of spam in our social bookmarking data sets in more detail, and we investigate the influence on recommendation performance.

[5]The data sets are available from http://ilk.uvt.nl/~toine/phd-thesis/.

3.2.1 CiteULike

CiteULike is a social bookmarking service that offers a "a free service to help you to store, organise, and share the scholarly papers you are reading"[6]. It allows its users to add their academic reference library to their online profiles on the CiteULike website. CiteULike was created in November 2004 by Richard Cameron and contains over 1.5 million unique articles[7], annotated by more than 46,000 users with over 307,000 unique tags at the time of writing. Articles can be stored with their metadata (in various formats), abstracts, and links to the papers at the publishers' sites. In addition, users can add reading priorities, personal comments, and tags to their papers. CiteULike also offers the possibility of users setting up and joining groups that connect users sharing academic or topical interests. These group pages report on recent activity, and offer the possibility of maintaining discussion fora or blogs. The full text of articles is not accessible from CiteULike, although personal PDF versions can be uploaded, which are then only accessible to the uploading user. Figure 3.1 shows a screenshot of a user's profile page in CiteULike.

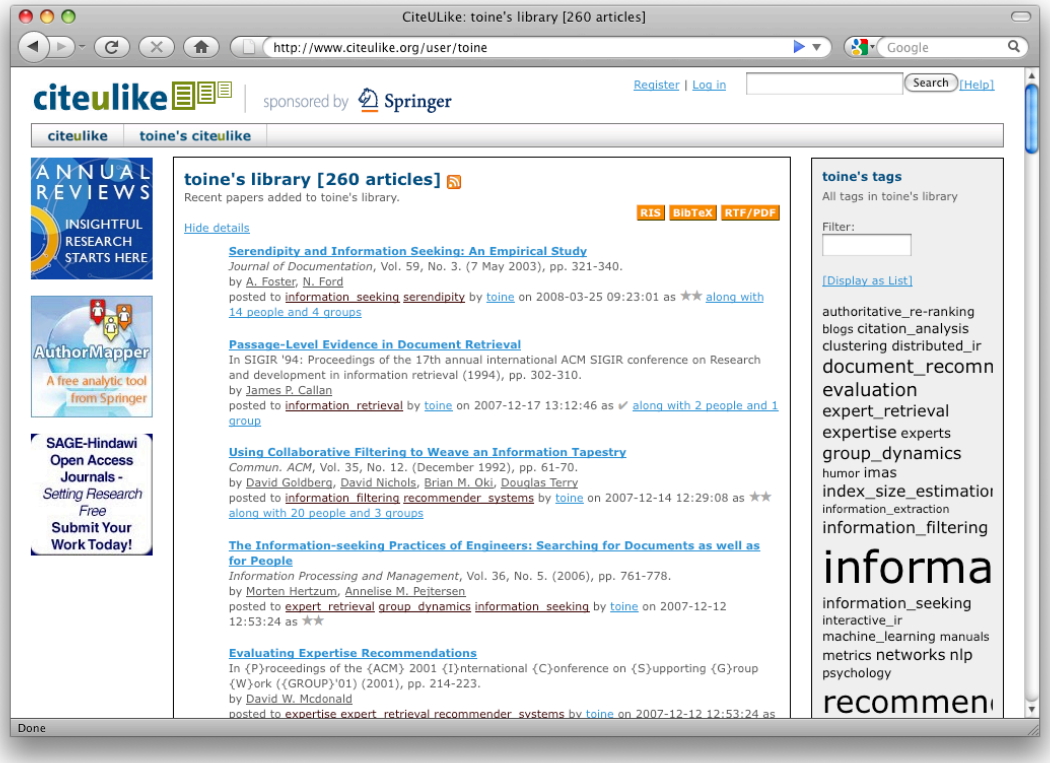


Figure 3.1: Screenshot of a user's profile page in CiteULike.

CiteULike offers daily dumps of their core database to benefit researchers wishing to analyze or experiment with the data[8]. We used the dump of November 2, 2007 as the basis for our experiments. A dump contains all information on which articles were posted by whom, with which tags, and at what point in time. Each line represents a user-item-tag triple with

[6] http://www.citeulike.org

[7] When we refer to items as articles, we mean not just journal articles, but also conference papers, theses, technical reports, books, and so on.

[8] Available from http://www.citeulike.org/faq/data.adp.

associated timestamp of the posting. These dumps, however, do not contain any of the other metadata described above. Because we want to investigate the usefulness of metadata in Chapter 5 we crawled this metadata ourselves from the CiteULike website using the article IDs. Appendix A contains more information about this crawling process and some of the problems we had to solve. We ended up collecting the following five types of metadata.

Topic-related metadata including all the metadata descriptive of the article's topic, such as the title and the publication information.

Person-related metadata such as the authors of the article as well as the editors of the journal or conference proceedings in which it was published.

Temporal metadata such as the year and, if available, month of the article's publication.

Miscellaneous metadata such as the article type. The extracted data also includes the publisher details, volume and number information, and the number of pages. DOI and ISSN/ISBN identifiers were also extracted as well as URLs pointing to the online whereabouts of the article.

User-specific metadata including the tags assigned by each user, comments by users on an article, and reading priorities.

A total of 803,521 articles were present in our November 2007 data dump. However, metadata could be crawled for only 585,351 (or 72.8%) of these articles. To make for a fair comparison between the recommendation algorithms that do and do not use metadata, we removed all articles from our data set for which we could not crawl metadata. This left us with a data set containing 18,072 user profiles and 585,351 articles. Here, the majority of the articles with missing metadata were spam entries, so this takes care of our spam filtering problem to a large extent. We refer the reader to Chapter 7 for we consider to be spam in social bookmarking systems, and more information on how we know the majority of articles with missing metadata to be spam.

3.2.2 BibSonomy

BibSonomy is a social bookmarking service for sharing bookmarks and reference lists of scientific articles[9]. It allows its users to add their academic reference library as well as their favorite bookmarks to their online profile on the BibSonomy website. Scientific articles are stored and represented as their BibTeX representation, and can include abstracts and links to the papers at the publishers' websites. Duplicate entries of resources already in the system are allowed as personal version of the resource, but are detected automatically by the system to improve browsing. Users can also describe their references using tags and use these to browse and discover new and related references (Hotho et al., 2006b). BibSonomy also allows its users to create a personal tag hierarchy by supplying the system with information about *is-a* relationships between tags. Figure 3.2 shows a screenshot of a

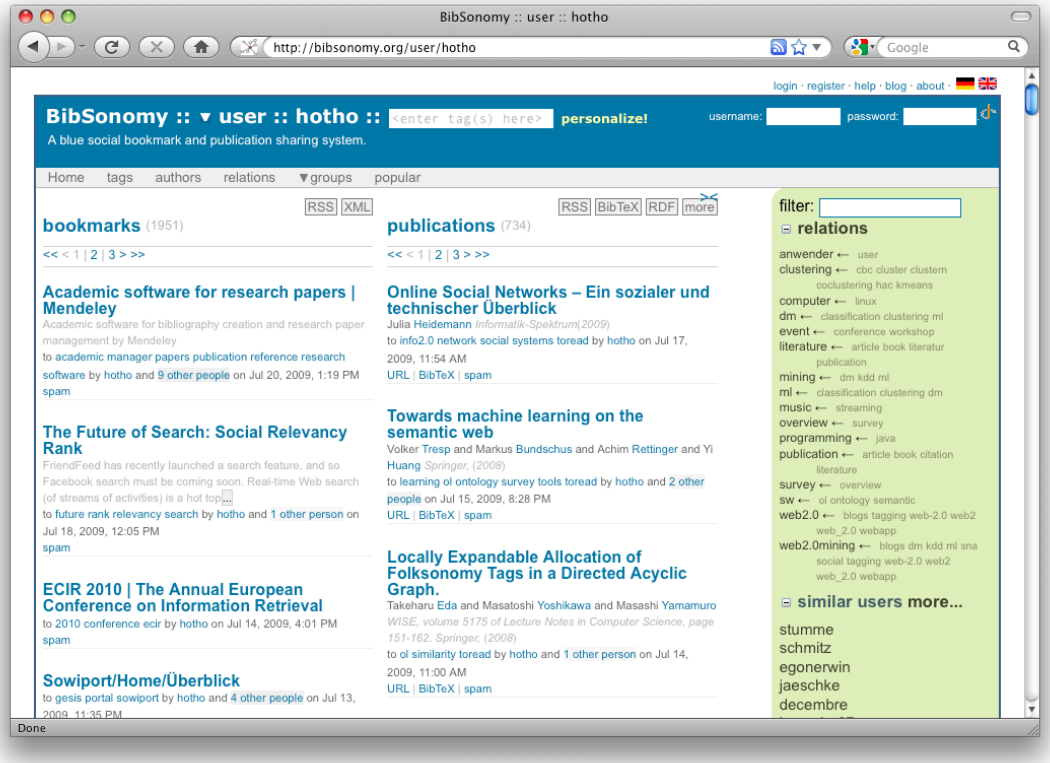


Figure 3.2: Screenshot of a user's profile page in BibSonomy.

user's profile page in BibSonomy; the tag hierarchy for this user is visible in the frame on the right-hand side.

BibSonomy is used as a testbed for research into various knowledge-organizational aspects of social bookmarking by the Knowledge and Data Engineering group of the University of Kassel, Germany. Some of the tasks that have been investigated using BibSonomy are constructing topic-specific rankings of bookmarked items (Hotho et al., 2006a), ontology learning from the BibSonomy folksonomy (Schmitz et al., 2006), tag recommendation for new posts (Jäschke et al., 2007b), the network properties of the folksonomy (Schmitz et al., 2007), and spam detection in social bookmarking (Krause et al., 2008). As part of their ongoing research efforts, they organized the 2008 ECML/PKDD Discovery Challenge which focused on social bookmarking. The BibSonomy data set was released to the public in May 2008 as part of this challenge[10]. Each year a different domain and problem set is chosen as that year's Discovery Challenge, which is meant to encourage a collaborative research effort at knowledge discovery and data mining, and an emphasis on business problems and solutions to those problems. In 2008, two tasks were selected based on the BibSonomy social bookmarking service: tag recommendation and spam detection. We go into more detail about the latter task in Chapter 7. For these tasks the organizers made a snapshot of their BibSonomy system available in the form of a MySQL dump. This dump consisted of all resources posted to BibSonomy between its inception in 2006 and March 31, 2008, and contains the same type of metadata for the scientific articles that we collected for

[9] http://www.bibsonomy.org/

[10] The website for the challenge can be found at http://www.kde.cs.uni-kassel.de/ws/rsdc08/

CiteULike. The distinction between bookmarks and BibTeX records is also made in this snapshot. We therefore split this data dump into a data set containing only Web bookmarks and a data set containing only scientific articles. Since it is possible and even encouraged by the BibSonomy service to add both bookmarks and articles to a user profile, this means that certain users can end up in both data sets. However, the participation in both data sets is fairly disjoint in reality: only 22.1% of the users have added both bookmarks and articles.

As required for the spam detection task the data set came with information identifying all spam content in the BibSonomy system. The Discovery Challenge organizers were able to collect data of more than 2,600 active users and more than 36,000 spammers by manually labeling users. The data dump contained these labels that identify users as spammers or non-spammers. A large portion of all users were spammers, with almost 14 spam users for each genuine user. We used these labels to remove all spam content from our two data sets, leaving us with 1,913 user profiles containing Web page bookmarks and 1,310 user profiles containing scientific articles. One of the reasons that the two BibSonomy data sets are an order of magnitude smaller than the CiteULike and Delicious data sets is the fact that there was so much spam content in the original data dump.

3.2.3 Delicious

Delicious is a social bookmarking service for storing, sharing, and discovering Web bookmarks[11]. It was created in September 2003 by Joshua Schachter and had grown to 5.3 million users and 180 million posts by November 2008[12]. Delicious allows its users to save, tag, manage, and share Web pages, and enables easy browsing of other Web pages, tags, and users by offering separate pages for each URL, user, and tag on Delicious. Figure 3.3 displays a screenshot of a user's profile page in Delicious. It shows the user's bookmarks on the left-hand side with the user-generated titles and descriptions for each bookmark, as well as the tags assigned by the user. The numbers next to each bookmark signal how many people have added this Web page to Delicious. The user's tag cloud is visible on the right-hand side of the screen. Users also have the possibility of forming a social network with other users. Users can declare themselves 'fans' of other users by adding them to their personal network; reciprocal fandom by two users is considered as friendship in Delicious. Finally, with regard to duplicate detection, Delicious does a mild form of automatic duplicate detection by calculating MD5 hashes of the URLs. However, from our experience with crawling Delicious, this does not catch all duplicates, although we can provide no authoritative figure or source for this.

Unlike BibSonomy and CiteULike, Delicious does not offer a data dump of their databases, so we gathered our data set by crawling a subset of the Delicious website. Since our focus was on item recommendation for users, our focus in crawling was to collect a balanced, unbiased set of user profiles, i.e., the complete set of bookmarks a user had posted to Delicious. We started our data collection process with an earlier crawl performed in November 2006, starting from the list of the 100 most popular tags on Delicious and performing an outward

[11] http://www.delicious.com/

[12] According to http://blog.delicious.com/blog/2008/11/delicious-is-5.html

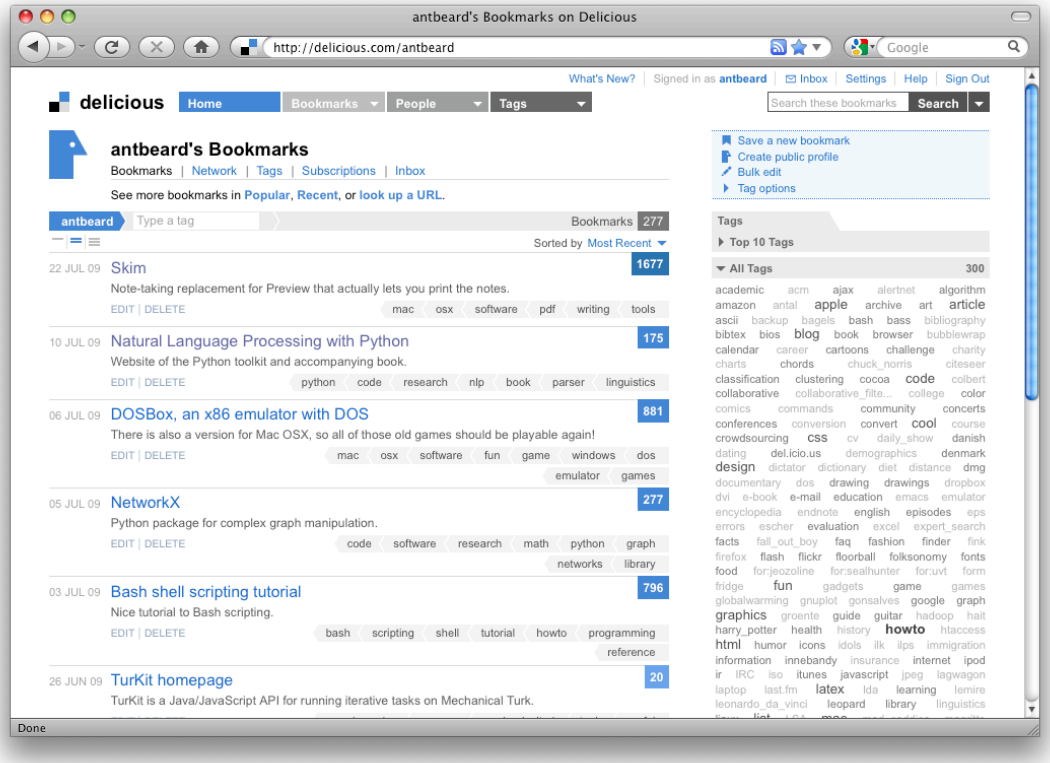


Figure 3.3: Screenshot of a user's profile page in Delicious.

crawl by treating Delicious as a tripartite graph. For each URL page, all new users and tags were added to the frontier, and for each user and tag page the frontier was likewise expanded with the new items, users, and tags. Selecting what tags, users, and items to crawl next was decided by ordering them by frequency, dividing the frequency range into 10 equal logarithmic bands, and then randomly selected an equal number of users, items, and tags from each frequency band. This crawl resulted in a list of 300,000 unique users. We randomly selected 100,000 users from this list to use as the starting point of crawling our actual data set. For each user we crawled their profile—the entire set of bookmarks posted by that user—and randomly selected about 18,000 of these users to match the size of our CiteULike data set. This final data set containing 17,975 user profiles is the one described in Table 3.1.

There are three potential problems with our data collection method. The first problem is that doing partial crawls of small-world graphs like Delicious can lead to a data set biased to the popular URLs, users, and tags (Heymann et al., 2008a). However, by selecting new targets to crawl by randomly selecting them from logarithmically divided frequency bands, we hope to have alleviated that to some extent. The second problem is that by randomly selecting users from a list of users already present on Delicious in November 2006 and then crawling their user profiles mid-2008, we observe that this precludes crawling any users who joined Delicious in 2007 or later. The third potential problem is that the crawled users have had more than a year to increase their profile, which will perhaps lead to a slight overestimation of certain statistics such as the average user profile size. However, we do not

believe that these problems will influence the outcome of our recommendation experiments to a great extent.

While we were able to filter out significant amounts of spam from our BibSonomy and CiteULike data sets, we do not have any comparable information about spam content in Delicious. However, (Heymann et al., 2008a) reported that for the data set they crawled from Delicious—in a similar manner to our crawled method—they found almost no spam. We therefore assume that spam presence on Delicious is not as big a problem as on BibSonomy and CiteULike.

3.3 Data Representation

Despite having different metadata fields, we have tried to keep our data representation as uniform as possible across domains and collections from the three different social bookmarking websites we focus on. We use two different formats to represent each data set: one to represent the *usage* information and another to represent the *metadata*. Usage information tells us what user added which items with which tags at what point in time. Our representation is similar to the way the public CiteULike data dumps are represented. Each post is represented as one or more lines: one for each tag assigned to the item by that user, represented by the user, item, time stamp, and tag. This representation is exactly the same for all collections. Figure 3.4 shows an example of how an example post from Delicious is represented.

USER ID	ITEM ID	TIMESTAMP	TAG
Edgecrusher7711	ba7544754ea353d2540733d0f43a5829	2008-02-28T17:50:00Z	hockey
Edgecrusher7711	ba7544754ea353d2540733d0f43a5829	2008-02-28T17:50:00Z	sports
Edgecrusher7711	ba7544754ea353d2540733d0f43a5829	2008-02-28T17:50:00Z	nhl

Figure 3.4: Example of the usage representation format.

In addition to usage data, we also have metadata available for the posts in our collection. Naturally, the amount of assigned metadata varies from user to user and item to item, but there is a systematic difference in the metadata fields available between the two domains of bookmarks (Web pages and scientific articles). The following metadata fields were available for the four different data sets.

BibSonomy bookmarks: <TITLE>, <DESCRIPTION>, and <URL>

Delicious: <TITLE>, <DESCRIPTION>, and <URL>

BibSonomy articles: <ABSTRACT>, <ADDRESS>, <AUTHOR>, <BOOKTITLE>, <CHAPTER>, <DAY>, <DESCRIPTION>, <EDITION>, <EDITOR>, <ENTRYTYPE>, <HOWPUBLISHED>, <INSTITUTION>, <JOURNAL>, <MONTH>, <NOTE>, <NUMBER>, <ORGANIZATION>, <PAGES>, <PUBLISHER>, <SCHOOL>, <SERIES>, <TITLE>, <TYPE>, <URL>, <VOLUME>, and <YEAR>

CiteULike: <ABSTRACT>, <ADDRESS>, <AUTHOR>, <BOOKTITLE>, <CHAPTER>, <EDITION>, <EDITOR>, <ENTRYTYPE>, <HOWPUBLISHED>, <INSTITUTION>, <JOURNAL>, <MONTH>, <NUMBER>, <ORGANIZATION>, <PAGES>, <PUBLISHER>, <SCHOOL>, <SERIES>, <TITLE>, <TYPE>, <URL>, <VOLUME>, and <YEAR>

These sets of metadata fields show that there is little difference between the available fields within domains. It is important to remark that we pre-process the <URL> field by replacing punctuation by whitespace, and by removing common prefixes and suffixes like www, http://, and .com. We use an SGML format to represent our data with a separate field for each metadata field. Figure 3.5 shows how the metadata of the example Delicious post from Figure 3.4 is represented in our SGML format.

```
<POST user="EdgeCrusher7711" item="ba7544754ea353d2540733d0f43a5829">
 <TITLE>
  The Fourth Period :: NHL Hockey Rumors, Trades, Breaking News and more
 </TITLE>
 <URL>
  http://www.thefourthperiod.com/
 </URL>
 <DESCRIPTION>
  -
 </DESCRIPTION>
</POST>
```

Figure 3.5: Example of the metadata representation format.

3.4 Experimental Setup

In order to evaluate different recommender algorithms on our four data sets and to compare the usefulness of the different information we have available, we need a proper framework for experimentation and evaluation. Recommender systems evaluation—and the differences with IR evaluation—has been addressed by, among others, Breese et al. (1998) and Herlocker et al. (1999, 2004), who make it abundantly clear that the choice for experimental setup and evaluation metrics is dependent on the recommendation task(s) a system is supposed to support. As mentioned earlier in Section 3.1, we focus on the *Find Good Items* task described by Herlocker et al. (2004), where users are provided with a ranked list of 'best bets' made by the system, based on their personal profile. Our experimental setup is therefore aimed at evaluating this recommendation task.

We start our description of the experimental setup by discussing how we filtered our raw data sets in Subsection 3.4.1. Then, in Subsection 3.4.2 we describe how we evaluated our experiments in terms of evaluation metrics and training-test splits. Subsection 3.4.3 critically discusses the choice we made in our experimental setup.

3.4.1 Filtering

It is common practice in recommender systems evaluation to select realistic subsets of the data sets used to ensure that reliable recommendations can be generated, and to allow for fair comparisons of different recommendation algorithms (Herlocker et al., 1999, 2004; McNee, 2006). This is typically done by filtering out users or items whose profile size or popularity falls below a certain threshold. It is common to retain only those users who have added 20 items or more to their personal profile (Herlocker et al., 1999, 2004). We follow this procedure in our preparation of the data sets as well. In addition, we filter out all items that occur only once, since these items do not contain sufficiently reliable ties to the rest of the data set, and thus would only introduce noise for recommendation algorithms[13]. Furthermore, we filter out all untagged posts from our data sets, because we want to investigate the influence tags can have on the recommendation process. By retaining untagged posts, it is more difficult to properly evaluate the performance of tag-based algorithms as users who never tag their posts would unfairly decrease the evaluation scores of the algorithms that rely on tags to generate their recommendations. The recommendation performance for the latter group of users can be extrapolated from the results of standard algorithms we test such as user-based and item-based CF.

Why Perform Filtering? The rationale for restricting our data set to users with 20 items or more in their profile is rooted in the fact that users need to have bookmarked at least some items before the system can generate useful recommendations for that user. Here, the 20-item-threshold has emerged in the literature as an accepted and realistic filtering setup in experiments, although there has been some research into the effect of sparse profiles on recommendation (see e.g., Breese et al. (1998) and Wang et al. (2006a)). Users new to the system could always be shown a list of the most popular content.

There is less consensus in the research community about the filtering threshold for *item* popularity, with thresholds reported in the literature ranging from 5 to 20 users minimum. The rationale behind setting such thresholds is that items with fewer or no ties to the rest of the data set will be more difficult to predict. In general, the higher the thresholds are set, the more dense the data set becomes. In turn, this reduces the set of possible items to choose from, making it easier to generate accurate predictions. This means that by reducing the size of the data set, performance is likely to go up, which would seem to be counterproductive to a proper evaluation. A second problem with stronger filtering is that it makes it impossible to dive into the long tail of items (Celma, 2008).

To ensure a realistic data set, we pick an item filtering threshold of two users and only throw out the *hapax legomena* items, i.e., the items that occur only once. Compared to the related work discussed in Subsection 2.1.1 and Section 4.5 our filtering regimen is among the most strict, i.e., for the sake of realism we aim to exclude as few users and items as possible. As a consequence, the recommendation task becomes more difficult, which is reflected in the scores. Moreover, we do not believe that filtering out the unique items detracts from the realism of our data sets. Although there is no reason to assume that the unique items are less

[13] If such filtering is done symmetrically, i.e., using the same filtering threshold for all node types, this is usually referred to as selecting the *p-core* of the data set, where p is the filtering threshold (Seidman, 1983; Jäschke et al., 2007b).

valuable to their users, we do believe that a considerable majority of these bookmarks are too personal to be of any collaborative interest. Exploration of our data sets has shown that these hapax items often include links to personal log-in pages or wish lists, which would not make for good recommendations to other users[14]. A third reason for filtering out the unique items has to do with our evaluation setup and how hapaxes are impossible to predict. This will be discussed in further detail in the next subsection.

Description of the Filtered Data Sets Table 3.2 shows the statistics of our four data sets after filtering, and is similar in structure to Table 3.1. These are the versions of our data sets that we will use in the experiments in the rest of this thesis, unless noted otherwise.

Table 3.2: Statistics of the filtered versions of our four data sets. Untagged posts were removed from each data set, and in these versions each user has at least 20 items in his profile and each item has been added by at least 2 people.

	bookmarks		articles	
	BibSonomy	**Delicious**	**BibSonomy**	**CiteULike**
# users	192	1,243	167	1,322
# items	11,165	152,698	12,982	38,419
# tags	13,233	42,820	5,165	28,312
# posts	29,096	238,070	29,720	84,637
avg # items per user	151.5	191.5	178.0	64.0
avg # users per item	2.6	1.6	2.3	2.2
avg # tags per user	203.3	192.1	79.2	57.3
avg # users per tag	2.9	5.6	2.6	2.7
avg # tags per item	8.4	4.8	3.1	5.3
avg # items per tag	7.1	17.0	7.7	7.3
max # items per user	2,404	2,451	6,325	1,264
max # users per item	41	66	35	188
max # tags per user	5,447	1,784	461	1,186
max # users per tag	117	737	97	299
max # tags per item	96	101	129	483
max # items per tag	1,461	12,021	4,497	1,603
user-item sparsity (%)	98.6427	99.8746	98.6291	99.8334

Table 3.2 shows a number of notable characteristics of the data sets. The two largest data sets we collected, Delicious and CiteULike, have about 1,300 users each after filtering, and the two BibSonomy data sets are a magnitude smaller. Users appear to be the most active on BibSonomy and Delicious with the average number of items in a user profile between two and three times as high as in the CiteULike collection. When we look at the number of tags assigned by the users, we see that users bookmarking Web pages tend to use the most tags: more than twice as many as users of social reference manager websites. We believe this reflects the greater topical diversity of Web bookmarks compared to scientific articles. In the latter scenario there may be many different research areas and topics, but in general the task and domain is pretty well-defined: academic research papers and articles. For social bookmarking websites, this would only be one possible domain they could cover. We claim that this greater topical diversity will be reflected later in performance: it is more

[14] This is especially true for the BibSonomy and Delicious bookmarks data sets.

difficult to predict for open domains than for closed domains. We revisit this notion later in Chapter 4.

All three social bookmarking services allow their users to tag items and the same items can be tagged by users other than the first user to post the item to the system. This means that the aggregation of all user tags forms a broad folksonomy for each of our four data sets. This is evident from Table 3.2: the average number of users per item is larger than 1 for all data sets. It is interesting to note that the average number of tags assigned per item is around two and a half times as large as the average number of users per item. This means that items are described to a greater extent by tags than they are by the users that have added them. This suggests that a recommendation algorithm that generates recommendations by comparing items could benefit from using tags instead of (or in addition to) just usage information, as the tag information is less sparse. In contrast, the average number of tags assigned by users is lower than or equal to the average number of items per user for three of our four data sets. This suggests that approaches that directly compare users may benefit less from using the assigned tags than from using the items added by those users.

3.4.2 Evaluation

We take a so-called *backtesting* approach to evaluation that is common in recommender systems literature (Breese et al., 1998; Herlocker et al., 2004; Baluja et al., 2008). For each user we withhold 10 randomly selected items from their profile, and generate recommendations by using the remaining data as training material. If a user's withheld items are predicted at the top of the ranked result list, i.e., if the algorithm is able to correctly predict the user's interest in those withheld items, then the algorithm is considered to perform well. In the rest of this thesis, we will refer to the user we are trying to recommend items for as the *active user* and his training set items as that user's *active items*.

This approach of withholding test items from user profiles also serves as an additional justification for removing the hapax items. When a hapax item is withheld for an active user, then it is not present in the training set anymore. If the recommendation algorithm does not know about the item from its presence in the training set, it can never be recommended to the active user. This puts an artificial ceiling on possible performance and makes for an unfair evaluation of the algorithms. If any item occurs at least twice in the entire data set, then the odds of that item being withheld for evaluation in both cases are much smaller, sketching a more realistic picture of performance. Naturally, the odds of items not being recommendable decrease as the item filtering threshold increases even further, the trade-off here being between unfair and unrealistic evaluation.

The performance of learning algorithms is typically influenced by several parameters. One way of optimizing these parameters is by maximizing performance on a given data set. Such tuning, however, tends to overestimate the expected performance of the system, so in order to prevent this kind of overfitting we use 10-fold cross-validation (Weiss and Kulikowski, 1991). For each of our four data sets, we first divide it into a training and a test set by randomly selecting 10% of the users to be in our test set. The final performance is evaluated on this 10% by withholding 10 items from each user, and using the remaining profile items

together with the training set to generate the recommendations for those 10%. In order to properly optimize parameters we divide our training set (containing 90% of the users in the entire collection) up again by randomly dividing the users over 10 folds, each containing 10% of the training users. Each fold is used as a validation set once, with 10 items being withheld for each validation fold user. The remaining profile items and the data in the other 9 folds are then used to generate the predictions. The final values for our evaluation metrics on the withheld items were then averaged over the 10 folds. This is comparable to macro evaluation in IR where the individual values for precision or recall are computed first and averaged afterwards. For our recommendation this means that the result is not biased towards the users with profiles that might be easier to predict. The goal in recommender systems research is to develop recommendation algorithms that can predict equally well for any user, not just the most active ones. Figure 3.6 visualizes this experimental setup.

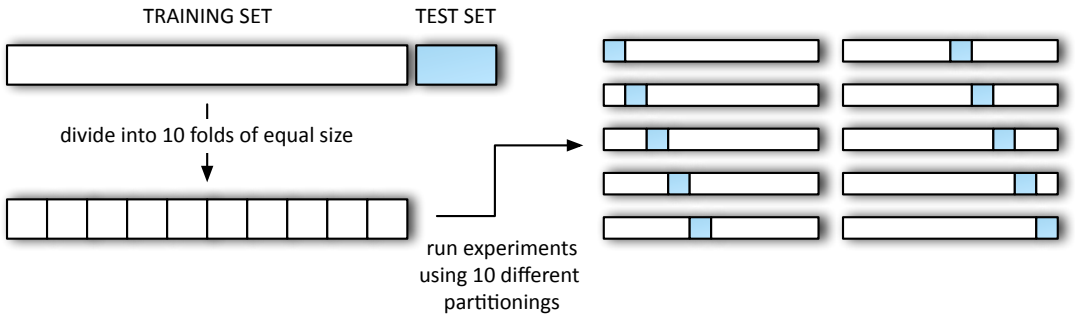


Figure 3.6: Visualization of our 10-fold cross-validation setup. First, the data set is split on user profiles into a training set and a test set (shaded). The training set is then further divided into 10 equal folds. In ten separate experiments, each fold serves as a test test once while the other nine folds serve as the training set. The results are then averaged over the ten test folds scores.

In our evaluation, we adopt an IR perspective by treating each of the users as a separate query. The 10 withheld items for each user make up the relevant items for which we have relevance judgments. For each user a ranked list of items is produced and evaluated on where these withheld items show up in the result list. While it is certainly possible and very likely that the recommendation lists contain recommendations that the user would find relevant or interesting, we cannot know this without the user judging them. This means that because our relevance judgments correspond to items added to a user's profile, we can never have any items judged as being not relevant without user intervention. This also means that the relevance judgments we derive from our experimental setup are unary and not binary: we only know if an item is relevant but never if it was not relevant.

Herlocker et al. (2004) assessed the usefulness of different metrics for each of the 6 recommendation tasks they identified. For our "Find Good Items" task they found that metrics that take into account the ranking of the items are most appropriate. We will therefore use Mean Average Precision (MAP) as our metric for evaluating the performance of our algorithms. MAP is one of the most popular metrics in IR evaluation and provides a stable "single-figure measure of quality across recall levels" (Manning et al., 2008). Average Precision is the average of the precision values calculated at each relevant retrieved item. To produce the

MAP, this value is then averaged over all queries. MAP rewards the retrieval of relevant items ahead of irrelevant items.

In addition to MAP, we evaluated the results of our experiments using three other metrics, two accuracy-based and one non-accuracy based. Mean Reciprocal Rank (MRR) is the average over all queries of the reciprocal of the rank of the first retrieved relevant item. Precision@N (P@N) is the precision at N retrieved items, i.e., the percentage of relevant items in the set of N retrieved items. The individual P@N values are also averaged over all queries to produce a single-figure measure, and we set N to 10. In our experiments we found that these two accuracy-based metrics painted the same picture of our algorithms as MAP did, with little or no variation in the statistical significance of differences in runs. We believe MAP to be more representative of performance than MRR or P@10, because of its emphasis on precision across different recall levels. For these reasons we only report MAP scores for our experiments; MRR or P@10 scores will only be mentioned if they help clarify the situation.

The fourth and final metric, coverage, pertains to an aspect of recommender systems evaluation that is not related to accuracy. Coverage is, depending on the definition, the percentage of items or users that the system can produce recommendations for (Herlocker et al., 2004). Certain recommendation algorithms need enough data on a user or item for them to be able to reliably generate recommendations. Not all algorithms will be able to cover every user or item. We measured *user coverage* (UCOV), which is the percentage of all users for which we were able to generate any predictions at all. As most of the algorithms we will discuss in this thesis have perfect user coverage, we will only report the UCOV measure when this is not the case for a certain algorithm and it is relevant for the discussion. For determining significance of differences between runs, we use a two-tailed paired T-test and report on significant differences using $^{\triangle}$ (and $^{\triangledown}$) for $\alpha = .05$ and $^{\blacktriangle}$ (and $^{\blacktriangledown}$) for $\alpha = .01$. For instance, the $^{\triangle}$ signals a statistically significant improvement over the baseline performance at an α of 0.05. Similarly, a $^{\blacktriangledown}$ signals a statistically significant decrease in performance compared to the baseline at an α of 0.01.

Breese et al. (1998) describe different experimental protocols that represent different levels of data-set sparsity. While the influence of sparsity on performance is an interesting issue, we leave the sparsity issue for future work. We focus only on the complete data sets in our experiments, i.e., we assume that, within our 10-fold cross-validation experimentation protocol, we can use all of the profile information we have available about the active users.

3.4.3 Discussion

All experimental setups for evaluating and comparing recommendation algorithms have their merits and their drawbacks. As the goal of a recommender system is to entice users to view, read, purchase, or otherwise use items that are new to them and of potential interest, the most direct and appropriate way to evaluate an algorithm is to run it in live service and measure the acceptance rate by real users. This is commonly regarded as one side of the evaluation spectrum. However, in a scenario where many different algorithms need to be compared, each with their own parameter settings, it is not always practical to explore all

these variations. Each variation would have to be run long enough and with sufficient users to be able to draw statistically significant conclusions about the differences in performance. Additionally, testing poorly performing variations could result in unacceptable degradation of the live performance of the systems for a subset of the user population (Baluja et al., 2008).

On the other end of the evaluation spectrum, backtesting and other offline evaluation methods make it easy to rapidly prototype and compare different algorithms, without having to rely on or wait for a sufficiently large group of users to work with the system and react to the recommendations. As it is still unclear which recommendation algorithms will perform best in social bookmarking systems, we first need to whittle down the space of possible variations before a live evaluation or comparison with real users should be attempted. For this reason, we focus on backtesting evaluation in this thesis. There are a number of issues with our system-based evaluation regimen that should be mentioned to put our results in the right perspective.

One weakness of offline analyses such as our system-based evaluation is that we are limited to an objective evaluation of our results, and based on this we cannot determine whether users will prefer one particular algorithm over another algorithm (Herlocker et al., 2004). System acceptance is dependent not just on prediction quality, but also on other more subjective criteria such as the user interface. A second problem of offline evaluation that was already discussed earlier, is that we only possess unary relevance judgments, which do not allow us to make any inferences about whether or not unseen items would be appreciated by the user. In addition, because backtesting is done on a snapshot of the social bookmarking system's database, it hard to simulate the dynamic nature of live recommendation, where new users and items are added continuously, with new trends and old trends coming and going. Such aspects are harder to capture in an offline analysis.

Finally, backtesting is a quite conservative paradigm for evaluation, and is likely to underestimate real performance for actual users, since we can only quantitatively evaluate the 10 withheld items for each user (Baluja et al., 2008). Recommendation on such large data sets is also a difficult task, since we are trying to pick the correct 10 withheld items from item sets ranging in size from 11,000+ tot 152,000+ items. This boils down to correctly picking out 0.09% of all items as the recommended items, which is like to trying to find a needle in a haystack. These are two of the main reasons that MAP values can be quite low overall, as we will see later. Ideally, our algorithms should not just be biased towards finding exactly what the user is missing, but rather they should find all useful recommendations.

Chapter 4

Folksonomic Recommendation

One of the defining characteristics of any social bookmarking system that supports social tagging, is the emergence of a folksonomy. This collaboratively generated categorization of items in a system serves to bind users and items together through an extra annotation layer. This extra layer of information can have many different applications, as discussed in Chapter 2. For instance, both searching and browsing through the content in a system can be improved by using the tags assigned to items by users. This leads us to our first research question.

RQ 1 How can we use the information represented by the folksonomy to support and improve the recommendation performance?

More specifically, we are interested in how we can improve the performance of collaborative filtering algorithms, which base their recommendations on the opinions or actions of other like-minded users as opposed to item content. We have two options for incorporating the tagging information contained in the entire folksonomy in CF algorithms. The first option is to treat the tag layer as an extra, refined source of usage information on top of the normal bipartite graph of users and items. Here, we take a standard Collaborative Filtering (CF) algorithm as our starting point and examine a number of ways that tags can be used to find like-minded users, or to re-rank the recommendations. We will refer to this as Tag-Based Collaborative Filtering (TBCF). The second option is to use a CF algorithm designed to operate on the entire tripartite graph. We will refer to such approaches as Graph-Based Collaborative Filtering (GBCF). We discuss the most important related work on GBCF approaches and compare two of the state-of-the-art GBCF algorithms to our own tag-based algorithms.

This chapter is organized as follows. We start in Section 4.1 by establishing the notation and definitions that we will use throughout this chapter. In Section 4.2 we establish a popularity-based recommendation baseline that is currently employed by several social bookmarking websites. Section 4.3 establishes a stronger baseline by applying two CF algorithms to our data sets. We examine a number of ways in which we can incorporate tags into our CF

algorithms and present these TBCF algorithms in Section 4.4. Section 4.5 discusses the related work. on recommendation for social bookmarking websites. In Section 4.6 we look at two state-of-the-art GBCF approaches from the related work and compare them to our own TBCF algorithms. In Section 4.7 we answer RQ 1 and present our conclusions.

4.1 Preliminaries

We start by establishing the notation and definitions that will be used throughout this chapter. To be consistent with other work on recommendation for social bookmarking, we base our notation in part on the work by Wang et al. (2006a), Clements et al. (2008a), and Tso-Sutter et al. (2008). In the social bookmarking setting that we focus on in this thesis, users post items to their personal profiles and can choose to label them with one or more tags. We recall that, in Chapter 2, we defined a folksonomy to be a tripartite graph that emerges from this collaborative annotation of items. The resulting ternary relations that make up the tripartite graph can be represented as a 3D matrix, or third-order tensor, of users, items, and tags. The top half of Figure 4.1 illustrates this matrix view. We refer to the 3D matrix as $\mathbf{D}(u_k, i_l, t_m)$. Here, each element $d(k,l,m)$ of this matrix indicates whether user u_k (with $k = \{1,\ldots,K\}$) tagged item i_l (with $l = \{1,\ldots,L\}$) with tag t_m (with $m = \{1,\ldots,M\}$), where a value of 1 indicates the presence of the ternary relation in the folksonomy.

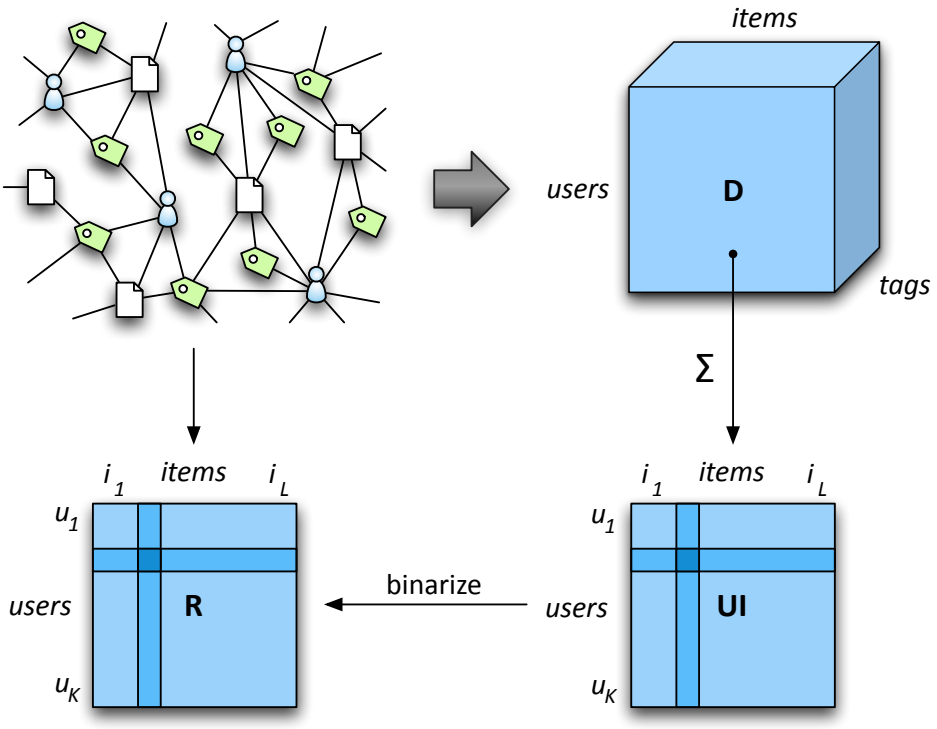


Figure 4.1: Representing the tripartite folksonomy graph as a 3D matrix. The ratings matrix **R** is derived from the tripartite graph itself and directly represents what items were added by which users. Aggregation over the tag dimension of **D** gives us matrix **UI**, containing the tag counts for each user-item pair. By binarizing the values in **UI** we can obtain **R** from **UI**. The figure is adapted from Clements et al. (2008a) and Tso-Sutter et al. (2008).

In conventional recommender systems, the user-item matrix contains rating information. The ratings can be *explicit*, when they are entered directly by the user, or *implicit*, when they are inferred from user behavior. In our case we have implicit, unary ratings where all items that were added by a user receive a rating of 1. All non-added items do not have a rating and are indicated by $\emptyset$. We extract this ratings matrix $\mathbf{R}(u_k, i_l)$ for all user-item pairs directly from the tripartite graph. We denote its individual elements by $x_{k,l} = \{1, \emptyset\}$. As evident from Figure 4.1, we can also extract a user-item matrix from $\mathbf{D}$ by aggregating over the tag dimension. We then obtain the $K \times L$ user-item matrix $\mathbf{UI}(u_k, i_l) = \sum_{m=1}^{M} \mathbf{D}(u_k, i_l, t_m)$, specifying how many tags each user assigned to each item. Individual elements of $\mathbf{UI}$ are denoted by $x_{k,l}$. We can define a binary version of this matrix $\mathbf{UI}_{binary}(u_k, i_l)$ as $sgn \sum_{m=1}^{M} \mathbf{D}(u_k, i_l, t_m)$ where the *sgn* function sets all values > 0 to 1. Because we filtered our data sets to include only tagged content, as described in Chapter 3, our ratings matrix $\mathbf{R}$ is the same as $\mathbf{UI}_{binary}$[1].

Similar to the way we defined $\mathbf{UI}$ we can also aggregate the content of $\mathbf{D}$ over the user and the item dimensions. We define the $K \times M$ user-tag matrix $\mathbf{UT}(u_k, t_m) = \sum_{l=1}^{L} \mathbf{D}(u_k, i_l, t_m)$, specifying how often each user used a certain tag to annotate his items. Individual elements of $\mathbf{UT}$ are denoted by $y_{k,m}$. We define the $L \times M$ item-tag matrix $\mathbf{IT}(i_l, t_m) = \sum_{k=1}^{K} \mathbf{D}(u_k, i_l, t_m)$, indicating how many users assigned a certain tag to an item. Individual elements of $\mathbf{IT}$ are denoted by $z_{l,m}$. We can define binary versions of $\mathbf{UT}$ and $\mathbf{IT}$ in a manner similar to $\mathbf{UI}_{binary}$. Figure 4.2 visualizes the 2D projections in the users' and items' tag spaces.

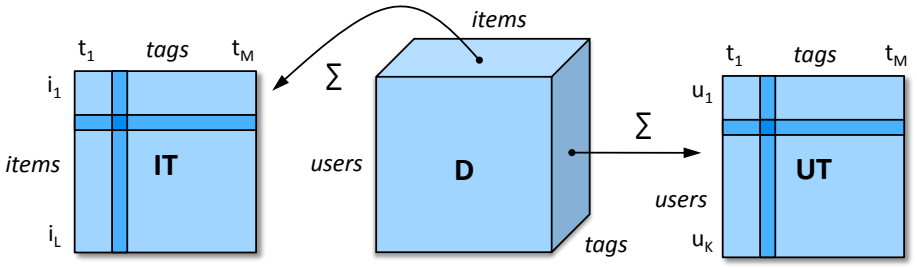


Figure 4.2: Deriving tagging information at the user level as **UT**, and the item level as **IT**, by aggregating over the item and user dimensions respectively.

The ratings matrix $\mathbf{R}$ can be represented by its row vectors:

$$\mathbf{R} = [\overrightarrow{u_1}, \ldots, \overrightarrow{u_k}]^T, \ \overrightarrow{u_k} = [x_{k,1}, \ldots, x_{k,L}]^T, \ k = 1, \ldots, K,$$

where each row vector $\overrightarrow{u_k}^T$ corresponds to a user profile, which represents the items that user added to his profile. $\mathbf{R}$ can also be decomposed into column vectors:

$$\mathbf{R} = [\overrightarrow{i_1}, \ldots, \overrightarrow{i_l}], \ \overrightarrow{i_l} = [x_{1,l}, \ldots, x_{K,l}]^T, \ l = 1, \ldots, L,$$

where each column vector $\overrightarrow{i_l}$ represents an item profile, containing all users that have added that item. We can decompose the $\mathbf{UI}$, $\mathbf{UT}$, and $\mathbf{IT}$ matrices in a similar fashion. We

[1] Note that this would not be the case if $\mathbf{R}$ contained explicit, non-binary ratings.

will also refer to the user and item profiles taken from the **UI** matrix as $\overrightarrow{u_k}$ and $\overrightarrow{i_l}$. We decompose the **UT** and **IT** matrices, which have the same number of columns M, into row vectors in the following way:

$$\mathbf{UT} = [\overrightarrow{d_1}, \ldots, \overrightarrow{d_k}]^T,\ \overrightarrow{d_k} = [y_{k,1}, \ldots, y_{k,M}]^T,\ k = 1, \ldots, K$$

$$\mathbf{IT} = [\overrightarrow{f_1}, \ldots, \overrightarrow{f_l}]^T,\ \overrightarrow{f_l} = [z_{l,1}, \ldots, z_{l,M}]^T,\ l = 1, \ldots, L$$

The vectors $\overrightarrow{d_k}$ and $\overrightarrow{f_l}$ are the tag count vectors for the users and items respectively.

Formally, the goal of each of the recommendation algorithms discussed in this chapter is to rank-order all items that are not yet in the profile of the active user u_k (so $x_{k,l} = \emptyset$) so that the top-ranked item is most likely to be a good recommendation for the active user. To this end, we predict a rating or score $\widehat{x}_{k,l}$ for each item that user u_k would give to item i_l. In our social bookmarking scenario we do not have explicit ratings information (e.g., 4 out of 5 stars), so we try to predict whether a user will like an item or not. The final recommendations for a user $RECS(u_k)$ are generated by rank-ordering all items i_l by their predicted rating $\widehat{x}_{k,l}$ as follows:

$$RECS(u_k) = \{\, i_l \mid rank\ \widehat{x}_{k,l},\ x_{k,l} = \emptyset \,\}. \tag{4.1}$$

Only items not yet in the user's profile $\overrightarrow{u_k}$ are considered as recommendations ($x_{k,l} = \emptyset$). Regardless of how the ratings $\widehat{x}_{k,l}$ are predicted, we always generate the recommendations according to Equation 4.1.

4.2 Popularity-based Recommendation

One of the most straightforward recommendation strategies is to recommend the most popular content in a system to every user. Such a popularity-based recommendation algorithm ranks the items in the system by popularity, and presents every user with this list of recommendations, minus the items the user already owns. Recommending items based solely on popularity in the system without any regard for personal preferences can be expected to produce poor recommendations. The list of most popular items reflects the combined tastes of all users in the system, and thus represents the 'average' user of the system. Rather few actual users will be similar to this average user.

In general, most users have only a few of the most popular items in their profile. The majority of their items, however, are in the long tail of the item distribution, and not in the list of most popular content. As a popularity-based algorithm cannot reach into the long tail, it cannot be expected to provide novel recommendations. We therefore consider popularity-based recommendations to serve as a weak baseline. We include the algorithm in this chapter, because popularity-based recommendations have become a standard feature

of many social news and social bookmarking websites, and therefore a staple algorithm to compare new algorithms against (Nakamoto et al., 2008; Zanardi and Capra, 2008; Wetzker et al., 2009). Delicious is one example of such a social bookmarking website that uses popularity-based recommendation[2]. We therefore include it as our weak baseline to show how much more sophisticated algorithms can improve upon a popularity-based algorithm.

Formally, we define popularity-based recommendation as calculating a normalized popularity score $\widehat{x}_{k,l}$ for each item according to Equation 4.2:

$$\widehat{x}_{k,l} = \frac{\left|\{ \overrightarrow{i_l} \mid x_{a,l} \neq \emptyset \}\right|}{K}, \tag{4.2}$$

where the total number of users that have added item i_l is counted and normalized by dividing it by the total number of users K in the data set. Table 4.1 contains the results of popularity-based recommendation runs on our four data sets. We reiterate here for the sake of the reader that we refer to the BibSonomy data sets containing scientific articles and Web bookmarks as BibArt and BibBoo respectively. For the sake of convenience we include them once in Table 4.1 as well.

Table 4.1: Results of the popularity-based baseline. Reported are the MAP scores.

Run	bookmarks		articles	
	BibSonomy (BibBoo)	**Delicious** (Delicious)	**BibSonomy** (BibArt)	**CiteULike** (CiteULike)
Popularity baseline	0.0044	0.0022	0.0122	0.0040

As expected, the results show that popularity-based recommendation does not achieve very high scores. Popularity-based recommendation achieves its highest MAP scores on the two smallest data sets, BibBoo and BibArt, and the lowest scores are achieved on the Delicious data set. However, all scores are unlikely to be of practical value for most users however. Popularity-based recommendation achieves its highest P@10 score on BibArt at 0.0143, which means that on average only 0.14 correct items can be found among the top 10 of recommended items. The highest MRR rank score on BibArt is 0.0565. This means that, on average, the first relevant withheld item is found at rank 18. The lowest MRR score on the Delicious data set means that the first relevant item there is found at rank 51 on average. Most users are unlikely to go through the first 50 recommended items to find only one that would be of value to them.

The MAP scores on the CiteULike data set are high in comparison to Delicious, the other large data set. It is easier for a popularity-based algorithm to deal with data sets with a smaller number of total items. With less items in total, the probability of a user's item to be part of the fixed-size set of most popular items is larger. Another pattern we see is that the scores on the data sets covering scientific articles are higher than those on the bookmarks data sets at corresponding data set sizes. We believe this is a reflection of the greater topical

[2]According to http://www.seomoz.org/blog/reddit-stumbleupon-delicious-and-hacker-news-algorithms-exposed (last visited: July 29, 2009).

diversity of social bookmarking websites over social reference managers as we mentioned earlier in Subsection 3.4.1. The greater diversity is reflected in a more difficult prediction task with lower scores.

4.3 Collaborative Filtering

A common and well-understood source of information for recommendations are usage patterns: who added or rated what content in the system? As mentioned earlier in Chapter 2, the class of algorithms that operate on such transaction data are called Collaborative Filtering algorithms (CF). We distinguished between two classes of CF algorithms—memory-based and model-based—and described the strengths and weaknesses of both types. In the experiments described in this chapter we focus on the k-Nearest Neighbor (k-NN) algorithm, one of the memory-based CF algorithms, and extend it in various ways to include tagging information in our TBCF algorithms. In this section we describe and establish the k-NN algorithm without tagging information as our strong baseline, and evaluate its performance on our four data sets. We implement and evaluate both the user-based and the item-based variants of the k-NN algorithm as introduced earlier in Subsection 2.1.1.

We pick the k-NN algorithm because it is a well understood algorithm that can intuitively be extended to incorporate other additional information (Herlocker et al., 1999; Burke, 2002). While model-based algorithms such as PLSA and matrix factorization have been shown to outperform memory-based algorithms in several cases (Hofmann, 2004; Koren, 2008), it is not always clear how to include extra information such as tags elegantly into these algorithms. Furthermore, memory-based algorithms better allow for the generation of intuitive explanations of why a certain item was recommended to the user. We see such functionality as an important component of social bookmarking systems, which rely heavy on user interaction and ease of use.

In the next Subsection 4.3.1 we formally define both variants of the k-NN algorithm. We present the results of the experiments with these algorithms on our data sets in Subsection 4.3.2, and discuss these results in Subsection 4.3.3.

4.3.1 Algorithm

The k-NN algorithm uses the behavior of similar users or items (the nearest neighbors) to predict what items a user might like, and comes in two 'flavors'. In *user-based filtering* we locate the users most similar to the active users, and then look among their items to generate new recommendations for the active users. In *item-based filtering* we locate the items most similar to items in the active user's profile, and order and present those similar items to the active user as new recommendations. In both cases the recommendation process is comprised of two steps: (1) calculating the similarity between the active object and other objects, and (2) using the N most similar neighboring objects to predict item ratings for

the active user[3]. In the first step we calculate the similarities between pairs of users or pairs of items. Many different similarity metrics have been proposed and evaluated over time, such as Pearson's correlation coefficient and cosine similarity (Herlocker et al., 1999; Breese et al., 1998). We use cosine similarity in our experiments because it has often been used successfully on data sets with implicit ratings (Breese et al., 1998; Sarwar et al., 2001).

User-based CF We first describe the user-based CF algorithm. User similarities are calculated on the user profile vectors $\overrightarrow{u_k}$ taken from the **R** matrix. We define the cosine similarity $sim_{cosine}(u_k, u_a)$ between two users u_k and u_a as

$$sim_{cosine}(u_k, u_a) = \frac{\overrightarrow{u_k} \cdot \overrightarrow{u_a}}{||\overrightarrow{u_k}|| \, ||\overrightarrow{u_a}||}. \tag{4.3}$$

The next step in user-based filtering is determining the top N similar users (or items) for user u_k. We denote this set as the Set of Similar Users $SSU(u_k)$ and define it as

$$SSU(u_k) = \{ u_a \mid rank\, sim_{cosine}(u_k, u_a) \leq N,\ x_{a,l} \neq \emptyset \}, \tag{4.4}$$

where we rank all users u_a on their cosine similarity $sim_{cosine}(u_k, u_a)$ to user u_k (or on another similarity metric), and take the top N. Consequently, $|SSU(u_k)| = N$. For each user u_a, we only consider those items that u_a added to his profile ($x_{a,l} \neq \emptyset$). The next step is to do the actual predictions for each item and generate the list of recommendations. The predicting score $\widehat{x}_{k,l}$ of item i_l for user u_k is defined as

$$\widehat{x}_{k,l} = \sum_{u_a \in SSU(u_k)} sim_{cosine}(u_k, u_a), \tag{4.5}$$

where the predicted score is the sum of the similarity values (between 0 and 1) of all N nearest neighbors that actually added item i_l (i.e., $x_{a,l} \neq \emptyset$). When applying user-based filtering to data sets with explicit ratings, it is common to scale $\widehat{x}_{k,l}$ to a rating in the original ratings scale (Sarwar et al., 2001). We do not scale our prediction $\widehat{x}_{k,l}$ as we are working with unary ratings and are only interested in rank-ordering the items by their predicted score[4].

A recurring observation from the literature about CF algorithms is that universally liked items are not as useful in capturing similarity between users as less common items, see e.g., Breese et al. (1998). Items that are added, rated, or purchased frequently can dominate the search for similar items, making it difficult to provide the user with novel recommendations. Adapted from the popular tf·idf term weighting algorithm from the field of IR (Salton and Buckley, 1988), we also try mitigating the influence of frequently occurring items by

[3]Note that since we reserved the letter k to index the users, we use N instead to denote the number of nearest neighbors.

[4]It is also common practice in CF to normalize the user's ratings by subtracting the user's average rating (Herlocker et al., 1999). However, we do not need to normalize in this manner, since we are working with implicit, unary ratings.

weighting the elements of $\overrightarrow{u_k}$ with the *inverse user frequency* of the user's items[5]. We define the inverse user frequency of item i_l as

$$idf(i_l) = \log \frac{K}{|\{ \overrightarrow{i_l} \mid x_{a,l} \neq \emptyset \}|}. \tag{4.6}$$

We then define user profile vectors $\overrightarrow{u'_k}$ weighted by the inverse user frequency as $\overrightarrow{u'_k} = [x_{k,1} \cdot idf(i_1), \ldots, x_{k,L} \cdot idf(i_L)]^T$. Similarity between two *idf*-weighted user vectors is also calculated using the cosine similarity. We will refer to these *idf*-weighted runs as U-IDF-SIM and to the runs without *idf*-weighting, which effectively use binary vectors, as U-BIN-SIM.

Item-based CF The item-based k-NN algorithm follows the same general principle as the user-based filtering algorithm. Instead of comparing users directly, we try to identify the best recommendations for each of the items in a user's profile. In other words, for item-based filtering we calculate the similarities between the test items of the active user u_k and the other items that u_k has not yet added (so $x_{k,b} = \emptyset$). Item similarities are calculated on the item profile vectors $\overrightarrow{i_l}$ taken from the **R** matrix. Similar to user similarity, we define cosine similarity $sim_{cosine}(i_l, i_b)$ between two items i_l and i_b as

$$sim_{cosine}(i_l, i_b) = \frac{\overrightarrow{i_l} \cdot \overrightarrow{i_b}}{||\overrightarrow{i_l}|| \, ||\overrightarrow{i_b}||}. \tag{4.7}$$

Analogous to user-based filtering, we can also suppress the influence of the most prolific users, i.e., users that have added a disproportionately large number of items to their profile, such as bots or spam users. This *inverse item frequency* of a user u_k is defined as

$$idf(u_k) = \log \frac{L}{|\{ \overrightarrow{u_k} \mid x_{k,b} \neq \emptyset \}|}. \tag{4.8}$$

We then define item profile vectors $\overrightarrow{i'_l}$ weighted by the inverse item frequency as $\overrightarrow{i'_l} = [x_{1,l} \cdot idf(u_1), \ldots, x_{K,l} \cdot idf(u_K)]^T$. Again, we calculate the similarity between two *idf*-weighted item vectors using cosine similarity. The next step is to identify the neighborhood of most similar items. We define the top N similar items as the Set of Similar Items $SSI(i_l)$

$$SSI(i_l) = \{ i_b \mid rank\, sim_{cosine}(i_l, i_b) \leq N,\ x_{k,b} \neq \emptyset \}, \tag{4.9}$$

where we rank all items i_b on their cosine similarity $sim_{cosine}(u_k, u_a)$ to item i_l (or on another similarity metric), and take the top N. For each item i_b, we only consider those items that are most similar to the items u_k added to his profile ($x_{k,b} \neq \emptyset$). The next step is to do the actual predictions for each item and generate the list of recommendations. The predicted score $\widehat{x}_{k,l}$ of item i_l for user u_k is defined as

[5] We refer to both the inverse user frequency and the inverse item frequency with idf for clarity and consistency with previous work.

$$\widehat{x}_{k,l} = \sum_{i_b \in SSI(i_l)} sim_{cosine}(i_l, i_b), \tag{4.10}$$

where the predicted score is the sum of the similarity values (between 0 and 1) of all the most similar items that were added by user u_k (i.e., $x_{k,b} \neq \emptyset$). The final recommendations $RECS(u_k)$ for user u_k for both algorithms are then generated as described in Section 4.1. We refer to these item-based CF runs with and without *idf*-weighting runs as I-IDF-SIM and I-BIN-SIM respectively. For the convenience of the reader, we have included a glossary in Appendix B that lists all of the run names used in Chapter 4 with a brief description.

Determining the Optimal Number of Nearest Neighbors After the user and item similarities are calculated, the top N neighbors are used to generate the recommendations. For k-NN classifiers, the neighborhood size is an algorithm parameter that can significantly influence prediction quality. Using too many neighbors might smooth the pool from which to draw the predictions too much in the direction of the items with the highest general popularity, whereas not considering sufficient neighbors might result in basing too many decisions on accidental similarities.

We use our 10-fold cross-validation setup to optimize the number of neighbors N as described in Subsection 3.4.2. The questions remains of what values of N should we examine? Examining all possible values from one single neighbor to considering all users or items as neighbors (where $N = K$ or L respectively) would be a form of overfitting in itself, as well as computationally impractical on the larger data sets. We therefore follow an iterative deepening approach as described by Van den Bosch (2004) for selecting the values of N. We construct a clipped, pseudo-quadratic series of 100 iterations. For each iteration q, the N value is determined by $N = 1.1^q$, rounded off to the nearest integer. As an example we list the first 40 unique values for N (corresponding to 55 iterations):

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 17, 19, 21, 23, 25, 28, 30, 34, 37,
41, 45, 49, 54, 60, 66, 72, 80, 88, 97, 106, 117, 129, 142, 156, 171, 189, ...

We employ the following heuristic optimization procedure to find the optimal number of neighbors. We keep doing additional iterations and evaluating values of N as long as the MAP score keeps increasing. We stop if the MAP score remains the same or decreases for a new value of N. To prevent the problem of ending up in a local maximum, we always take 5 more additional steps. If the MAP score has not increased statistically significantly anywhere in those 5 steps, then we take as our N the value corresponding to the maximum MAP score we encountered so far. If there is a significant increase in MAP within those 5 extra steps, we assume we found a local maximum and keep evaluating additional iterations. When we reach the maximum for N (the number of users or items in the data set, depending on the algorithm) we stop and take as N the value corresponding to the highest MAP score. For example, if the MAP score stops increasing at when $N = 13$, we examine the next 5 N values from 14 to 21. If none of those additional runs have MAP scores significantly higher than the MAP for $N = 13$, we take the N with the maximum MAP score in the interval $[13, 21]$.

4.3.2 Results

The relatively low scores achieved by the popularity-based algorithm of the previous section mean that there is much room for improvement. Table 4.2 shows the results of the two user-based and item-based variants of the k-NN algorithm. As expected, the k-NN algorithm outperforms popularity-based recommendation on all four data sets, with improvements ranging from a 109% increase on Delicious to a more than twenty-fold increase on CiteULike. These improvements are statistically significant on all data sets except Delicious.

Table 4.2: Results of the k-Nearest Neighbor algorithm. Reported are the MAP scores as well as the optimal number of neighbors N. Best-performing runs for each data set are printed in bold. The percentage difference between the best popularity-based run and the best CF run is indicated in the bottom row of the table.

Runs	bookmarks				articles			
	BibSonomy		Delicious		BibSonomy		CiteULike	
	MAP	N	MAP	N	MAP	N	MAP	N
Popularity baseline	0.0044	-	0.0022	-	0.0122	-	0.0040	-
U-BIN-SIM	0.0258▲	6	**0.0046**	15	**0.0865**▲	4	0.0746▲	15
U-BIN-IDF-SIM	**0.0277**▲	13	0.0040	15	0.0806▲	4	0.0757▲	15
I-BIN-SIM	0.0224▲	13	0.0027	25	0.0669▲	37	0.0826▲	117
I-BIN-IDF-SIM	0.0244▲	34	0.0025	14	0.0737▲	49	**0.0887**▲	30
% Change	+529.5%		+109.1%		+609.0%		+2117.5%	

User-based filtering outperforms item-based filtering on three of four data sets; only on CiteULike does item-based filtering work better, where this difference is also statistically significant ($p < 0.05$). The other differences between user-based and item-based filtering are not significant. There appears to be no clear advantage to applying *idf*-weighting to the profile vectors: there are small differences in both directions, but none of the differences between the runs with and without *idf*-weighting are significant. In general, it seems that bookmark recommendation is more difficult than article recommendation: even within the same collection, recommending BibSonomy bookmarks on BibBoo achieves MAP scores that are nearly three times as low as recommending BibSonomy articles using BibArt. While the BibArt and CiteULike performance numbers are about equal despite the size difference between the two data sets, the difference between BibBoo and Delicious is much larger. Finally, we see that the optimal number of neighbors tends to be larger for item-based filtering than for user-based filtering, but the actual number seems to be data set dependent.

4.3.3 Discussion

Our first observation is that memory-based CF algorithms easily outperform a recommendation approach based solely on popularity. This was to be expected as recommending items based solely on popularity in the system without any regard for personal preferences can be expected to produce poor recommendations. In contrast, CF learns the personal preferences of users, leading to better, more personalized recommendations. The user-based filtering algorithm achieved higher MAP scores on three of our four data sets, although these differences were not statistically significant. What could be the explanation for these

differences between user-based and item-based filtering? A possible explanation for this is that the average number of items per user is much higher than the average number of users per item. Since there are more items to potentially match, calculating a meaningful overlap between user profile vectors could be easier than between item profile vectors. This could also explain why item-based filtering works worst on Delicious, as it has the lowest average at 1.6 users per item.

The difference in the optimal number of neighbors between user-based and item-based can also be explained by this. The average number of users per item is much lower than the average number of items per users, which makes it more difficult to calculate meaningful overlap between item profile vectors. It is then understandable that the item-based filtering algorithm would need more nearest neighbors to generate correct predictions. However, the user profile vectors in our data sets are slightly more sparse than the item profile vectors, which would put user-based filtering at a disadvantage as there might be less overlap between the different users. In practice, these two forces are in balance, leading to the lack of significant differences between user-based and item-based filtering.

A second observation was that recommending bookmarks appears to be more difficult than recommending scientific articles: MAP scores on the article data sets are nearly three times as high as the MAP scores on the bookmarks data sets. We believe this reflects that bookmark recommendation is a more difficult problem because of the open domain. In our article recommendation task there may be many difficult research areas and topics, but in general the task and domain is pretty well-defined: academic research papers and articles. In contrast, the Delicious and BibBoo data sets cover bookmarked Web pages, which encompass many more topics than scientific articles tend to do. Users can be expected to have more different topics in their profile, making it more difficult to recommend new, interesting bookmarks based on their profiles.

To determine whether this explanation is correct, we need to show that user profiles containing Web bookmarks are topically more diverse that profiles containing only scientific articles. In all of our data sets we have topic representations of the content in the form of tags. Tags often represent the intrinsic properties of the items they describe, and we can use these tags to estimate how topically diverse the user profiles are in our four data sets. We can expect users with topically diverse profiles to have a smaller average tag overlap between their items. One metric that can be used to represent topical diversity is the average number of unique tags per user as reported in Table 3.2. These averages are 203.3 and 192.1 for BibBoo and Delicious respectively, which is significantly higher than the 79.2 and 57.3 for BibArt and CiteULike.

However, it is possible that Delicious and BibBoo users simply use more tags to describe their bookmarks on average than users who describe scientific articles, but still have a low topical diversity. To examine this, we calculate the average Jaccard overlap between the tags assigned to the item pairs for each user profile separately, and then generate a macro-averaged tag overlap score for each data set. This number represents an approximation of the topical diversity of a data set. We find that the average tag overlap for BibBoo and Delicious is 0.0058 and 0.0001 respectively, whereas for BibArt and CiteULike it is 0.0164 and 0.0072 respectively. This seems to be in line with our assumption that bookmark

recommendation is a more difficult task, because the user profiles are more diverse, and therefore harder to predict. The difference between performance on BibBoo and Delicious might be explained by the fact that BibSonomy, as a scientific research project, attracts a larger proportion of users from academia. In contrast, the scientific community is only a subset of the user base of Delicious, which could again lead to a greater diversity in topics.

4.4 Tag-based Collaborative Filtering

If we only applied the standard memory-based CF algorithms to our data sets, we would be neglecting the extra layer of information formed by the tags, which could help us produce more accurate recommendations. As mentioned before, we consider two options for incorporating the tags: (1) extending existing CF algorithms to create TBCF algorithms, or (2) recommending based on the entire tripartite graph (GBCF algorithms). In this section we propose three different TBCF algorithms, corresponding to three different ways of extending the standard k-NN algorithm to incorporate tag information and find like-minded users.

The first TBCF algorithm employs user and item similarities based on the overlap in tags assigned to items. We discuss this algorithm in Subsection 4.4.1. Subsection 4.4.2 describes the second type of TBCF algorithm, which calculate user and item similarities based on the overlap in tagging intensity, without looking at the actual overlap between tags. The third TBCF algorithm, described in Subsection 4.4.3, combines tagging information with usage information by fusing together (1) the similarities based on tagging overlap, from the first type of TBCF algorithm, and (2) the similarities based on usage information, i.e., from regular CF. We present the results of our experiments with these three TBCF algorithms in Subsection 4.4.4 and discuss the results in Subsection 4.4.5.

4.4.1 Tag Overlap Similarity

The folksonomies present in our four data sets carry with them an extra layer of connections between user and items in the form of tags. The tag layer can be used to examine other ways of generating similarities between users or items. For instance, users that assign many of the same tags and thus have more tag overlap between them, can be seen as rather similar. Items that are often assigned the same tags are also more likely to be similar than items that share no tag overlap at all. We propose calculating the user and item similarities based on overlap in tagging behavior as opposed to usage information that only describes what items a user has added. We will refer to a TBCF algorithm using tag overlap similarities for CF as TOBCF.

What do we expect from such an approach? In the previous section we signaled that sparsity of user profile and item profile vectors can be a problem for the standard memory-based CF algorithms. As users tend to assign multiple tags to an item—with averages ranging from 3.1 to 8.4 in our data sets—when they post it to their personal profile, this means a reduced sparsity, which could lead to better predictions. We expect this effect to be the strongest

for item-based filtering. On average, the number of tags assigned to an item is 2.5 times higher than the number of users who have added the item. This means that, on average, item profile vectors from the **IT** matrix are less sparse than item profile vectors from the **UI** matrix. This difference is not as well-pronounced for the items per user and tags per user counts: in some data sets users have more items than tags on average, and more tags than items in other data sets. This leads us to conjecture that using tags for user-based filtering will not be as successful.

Many different similarity metrics exist, but we restrict ourselves to comparing three metrics: Jaccard overlap, Dice's coefficient, and the cosine similarity. The only difference between this approach and the standard CF algorithm is in the first step, where the similarities are calculated.

User-based Tag Overlap CF For user-based TOBCF, we calculate tag overlap on the **UT** matrix or on the binarized version $\mathbf{UT}_{binary}$, depending on the metric. These matrices are derived as shown in Figure 4.2. Both the Jaccard overlap and Dice's coefficient are set-based metrics, which means we calculate them on the binary vectors from the $\mathbf{UT}_{binary}$ matrix. The Jaccard Overlap $sim_{Jaccard}(d_k, d_a)$ between two users d_k and d_a is defined as

$$sim_{UT-Jaccard}(d_k, d_a) = \frac{|\overrightarrow{d_k} \cap \overrightarrow{d_a}|}{|\overrightarrow{d_k} \cup \overrightarrow{d_a}|}. \tag{4.11}$$

Likewise, Dice's coefficient $sim_{Dice}(d_k, d_a)$ is defined as

$$sim_{UT-Dice}(d_k, d_a) = \frac{2|\overrightarrow{d_k} \cap \overrightarrow{d_a}|}{|\overrightarrow{d_k}| + |\overrightarrow{d_a}|}. \tag{4.12}$$

We refer to the user-based runs with Jaccard overlap and Dice's coefficient as UT-JACCARD-SIM and UT-DICE-SIM respectively. The cosine similarity is calculated in three different ways. First, we calculate it on the regular tag count vectors from **UT** as $sim_{UT-cosine-tf}(d_k, d_a)$, and on the binary vectors from the $\mathbf{UT}_{binary}$ matrix as $sim_{UT-cosine-binary}(d_k, d_a)$. In addition to this, we also experiment with *idf*-weighting of the tags in the user tag count vectors according to Equation 4.6. We then calculate the cosine similarity $sim_{UT-cosine-tfidf}(d_k, d_a)$ between these weighted user profile vectors. In each case, the cosine similarity is calculated according to

$$sim_{UT-cosine}(d_k, d_a) = \frac{\overrightarrow{d_k} \cdot \overrightarrow{d_a}}{||\overrightarrow{d_k}||\,||\overrightarrow{d_a}||}. \tag{4.13}$$

We refer to these three runs as UT-TF-SIM, UT-BIN-SIM, and UT-TFIDF-SIM respectively.

Item-based Tag Overlap CF For item-based TOBCF, we calculate the item-based versions of the similarity metrics $sim_{IT-Jaccard}(f_l, f_b)$, $sim_{IT-Dice}(f_l, f_b)$, and $sim_{IT-cosine-binary}(d_k, d_a)$ on $\mathbf{IT}_{binary}$. We calculate the tag frequency and weighted tag frequency vectors similarities

$sim_{IT-cosine-tf}(d_k, d_a)$ and $sim_{IT-cosine-tfidf}(d_k, d_a)$ on **IT**. We refer to these five item-based runs as IT-JACCARD-SIM, IT-DICE-SIM, IT-BIN-SIM, IT-TF-SIM, and IT-TFIDF-SIM respectively.

4.4.2 Tagging Intensity Similarity

Instead of looking at tag overlap as a measure of user or item similarity, we can also look at another aspect of tagging behavior to locate kindred users or items: *tagging intensity*. Adding a tag costs the user a small amount of effort, which could signal that users are more invested in those items they to which assign many tags. Tagging intensity could therefore be seen as an approximation to a user actively rating his items. Here, more effort invested in tagging corresponds to a higher rating for that item. If two users both assign many tags to the same items and only few tags to other items, they can be thought of as being more similar in tagging behavior.

Naturally, users might very well assign many different tags to an item for different reasons. Some items might simply be too complex to describe by just one or two tags. Furthermore, the assumption that more richly tagged items would also be rated more highly by the users is not without its caveats. Indeed, Clements et al. (2008a) investigated this for their data set based on the LibraryThing[6] data set. They compared actual book ratings to the number of tags assigned to the books and found only a weak positive correlation between the two. We can not repeat this, since we only have unary ratings in our data sets. However, the idea has its merits, since Clements et al. did not find any significant differences between using actual ratings for recommendation compared to using the tag counts.

We propose our own user (and item) similarity metrics that compare users (and items) on the intensity of their tagging behavior. We will refer to the algorithms that use these similarity metrics as TIBCF algorithms. The **UI** matrix contains the tag counts associated with each post in a data set. These represent how many tags were assigned to a certain item by a user. For user-based TIBCF we calculated these tag intensity-based similarities on the user profile vectors from **UI** according to Equation 4.3. For item-based TIBCF we calculate the item similarities on the item profile vectors from **UI** according to Equation 4.7. The rest of the approach follows the standard k-NN algorithm described in Subsection 4.3.1.

4.4.3 Similarity Fusion

The third TBCF algorithm we propose is one that combines two different algorithms: (1) the standard k-NN CF algorithm from Section 4.3, which uses data about all users' preferences for items to generate recommendations, and (2) the TOBCF algorithm from Subsection 4.4.1, which looks at the overlap in tagging behavior to identify the like-minded users and generate recommendations. As both approaches use different information to generate their recommendations, it is possible that an approach that combines the best of both worlds will outperform the individual approaches.

[6] http://www.librarything.com

There are many different ways of combining different approaches. We propose one possibility here, *similarity fusion* , where we linearly combine different sets of user similarities into a single set of similarities, as illustrated in Figure 4.3. The same principle holds for item-based filtering: there, we linearly combine the item similarities. We will refer to this approach as SimFuseCF.

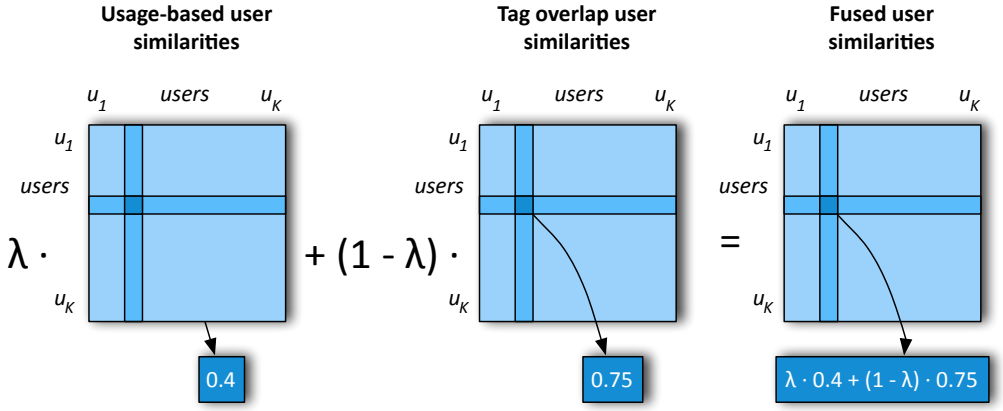


Figure 4.3: Fusing the usage-based and tag-based similarity matrices for user-based TBCF. The highlighted cells show how individual user-user similarities are combined. Similarity between a user and himself is always equal to 1.

In our SimFuseCF algorithm, we take as input the similarity matrices from two different approaches and linearly combine them element by element using a weighting parameter λ of which the value lies in the range [0,1]. The similarities were normalized into the $[0,1]$ range for each set of similarities separately. Fusing the two different similarity sources is done according to

$$sim_{fused} = \lambda \cdot sim_{usage} + (1-\lambda) \cdot sim_{tag-overlap}, \tag{4.14}$$

where sim_{usage} and $sim_{tag-overlap}$ are the usage-based and tag-based similarities for either user pairs or items pairs, depending on the choice for user-based or item-based filtering. By varying λ, we can assign more weight to one type of similarity or the other. The optimal λ value as well as the optimal number of neighbors N were optimized doing parameter sweeps using our 10-fold cross-validation setup. For each data set, we determined the optimal value of λ using a parameter sweep, examining all values between 0 and 1 with increments of 0.1. Setting λ to either 0 or 1 corresponds to using the similarities from the TOBCF or the CF algorithm respectively. Linearly combining the different similarity types is not the only possible combination approach. However, while more sophisticated feature combination and fusion approaches have been proposed for collaborative filtering in the past (Wang et al., 2006a), we leave a more principled combination of usage and tag information for future work.

We also considered two alternative fusion schemes. In the first, we combined usage-based similarities from standard CF with the similarities generated for the TIBCF algorithm in Subsection 4.4.2. The second fusion scheme combines the similarities from the TOBCF algorithm with the TIBCF algorithm, in effect combining the similarities based on tag overlap

and tagging intensity. Preliminary experiments suggested, however, that these two fusion schemes do not produce any acceptable results. We therefore restricted ourselves to fusing the usage-based similarities from CF with the tag overlap similarities from TOBCF.

4.4.4 Results

Below we present the results for our three algorithms based on tag overlap (TOBCF), tagging intensity (TIBCF), and similarity fusion (SimFuseCF).

Tag Overlap Similarity Table 4.3 shows the results of the user-based and item-based variants of the TOBCF algorithm. We consider the best-performing CF runs from Section 4.3 runs as the baseline runs and compare them to the TOBCF runs. What we see in Table 4.3 is that item similarities based on tag overlap work well for item-based filtering, as three of our four data sets show considerable improvements over the best CF baseline runs. Performance increases range from 27% on BibArt to almost 120% on Delicious, but these are only statistically significant on the Delicious data set. A first observation is the opposite trend for user-based filtering, where tag overlap results in significantly worse scores for almost all variants on all data sets, with performance decreases ranging from 40% to 63%. This means that using tag overlap in item-based filtering now makes item-based filtering outperform user-based filtering on all four data sets. A second observation is that on the bookmark collections tag overlap performs even worse for user-based filtering than on the data sets containing scientific articles. The reverse seems to be true for item-based filtering. It seems that the domain of the social bookmarking website influences the effectiveness of the TOBCF algorithm.

The results of the different tag overlap metrics tend to be close together and differences between them are not statistically significant. Even though the best performing metrics are dependent on the data set, we do see that the metrics operating on the binary vectors from the $\mathbf{UT}_{binary}$ and $\mathbf{IT}_{binary}$ matrices are among the top performing metrics. It is interesting to note that although that the runs with *idf*-weighting tend to perform worst of all five metrics, the IT-TFIDF-SIM produces the best results on the BibArt collection.

A final observation concerns the number of nearest neighbors N used in generating the recommendations. In general, the optimal number of neighbors is slightly lower than when using similarities based on usage data. For the item-based runs on BibBoo and BibArt the number of neighbors is lower than for user-based which is unexpected, given the neighborhood sizes of regular CF in Table 4.2.

Tagging Intensity Similarity Table 4.4 shows the results of the two user-based and item-based variants of the k-NN algorithm that use similarities based on tagging intensity. Tagging intensity does not appear to be representative of user or item similarity: 14 out of 16 runs perform worse than the best CF runs on those data sets, but most of them not significantly. It is interesting to note that tagging intensity similarities do produce small improvements in MAP scores on the user-based runs on Delicious. If we disregard these statistically not significant improvements, using tagging intensity as a source of user or item

Table 4.3: Results of the TOBCF algorithm. Reported are the MAP scores as well as the optimal number of neighbors N. Best-performing tag overlap runs for both user-based and item-based are printed in bold. The percentage difference between the best baseline CF runs and the best tag overlap runs are indicated after each filtering type.

Runs	bookmarks				articles			
	BibSonomy		Delicious		BibSonomy		CiteULike	
	MAP	N	MAP	N	MAP	N	MAP	N
Best UB CF run	0.0277	13	0.0046	15	0.0865	4	0.0757	15
	(U-BIN-IDF-SIM)		(U-BIN-SIM)		(U-BIN-SIM)		(U-BIN-IDF-SIM)	
UT-JACCARD-SIM	0.0070	8	0.0015	11	**0.0459**	6	**0.0449**$^{\blacktriangledown}$	5
UT-DICE-SIM	0.0069$^{\triangledown}$	6	0.0007$^{\blacktriangledown}$	6	0.0333$^{\triangledown}$	4	0.0439$^{\blacktriangledown}$	2
UT-BIN-SIM	**0.0102**	5	**0.0017**	11	0.0332$^{\triangledown}$	4	0.0452$^{\blacktriangledown}$	3
UT-TF-SIM	0.0069$^{\triangledown}$	2	0.0015$^{\triangledown}$	25	0.0368	4	0.0428$^{\blacktriangledown}$	8
UT-TFIDF-SIM	0.0018$^{\triangledown}$	6	0.0013$^{\triangledown}$	17	0.0169$^{\blacktriangledown}$	2	0.0400$^{\blacktriangledown}$	2
% Change over best UB CF run	-63.2%		-63.0%		-46.9%		-40.7%	
Best IB CF run	0.0244	34	0.0027	25	0.0737	49	0.0887	30
	(I-BIN-IDF-SIM)		(I-BIN-SIM)		(I-BIN-IDF-SIM)		(I-BIN-IDF-SIM)	
IT-JACCARD-SIM	**0.0370**	3	0.0083$^{\triangle}$	21	0.0909	6	0.0810	14
IT-DICE-SIM	0.0317	2	0.0089$^{\triangle}$	25	0.0963	8	**0.0814**	8
IT-BIN-SIM	0.0334	2	**0.0101**$^{\triangle}$	23	0.0868	5	0.0779	10
IT-TF-SIM	0.0324	4	0.0100$^{\triangle}$	11	0.0823	4	0.0607$^{\blacktriangledown}$	17
IT-TFIDF-SIM	0.0287	8	0.0058	7	**0.1100**	7	0.0789	21
% Change over best IB CF run	+51.6%		+274.1%		+49.3%		-8.2%	
% Change over best CF run	+33.6%		+119.6%		+27.2%		-8.2%	

similarity decreases performance by around 20% on average. In fact, the performance of item-based filtering on Delicious is even worse than the popularity-based algorithm.

Table 4.4: Results of the TIBCF algorithm. Reported are the MAP scores as well as the optimal number of neighbors N. Best-performing tagging intensity runs for each data set are printed in bold. The percentage difference between the best baseline CF run and the best CF run with tagging intensity similarities is indicated in the bottom row of the table.

Runs	bookmarks				articles			
	BibSonomy		Delicious		BibSonomy		CiteULike	
	MAP	N	MAP	N	MAP	N	MAP	N
Best CF run	0.0277	13	0.0046	15	0.0865	4	0.0887	15
	(U-BIN-IDF-SIM)		(U-BIN-SIM)		(U-BIN-SIM)		(I-BIN-IDF-SIM)	
U-TF-SIM	0.0229	13	**0.0061**	60	**0.0711**	11	0.0700$^{\blacktriangledown}$	14
U-TFIDF-SIM	**0.0244**	8	0.0052	45	0.0705	9	0.0709$^{\blacktriangledown}$	37
I-TF-SIM	0.0179	21	0.0004$^{\blacktriangledown}$	10	0.0624	17	0.0774$^{\blacktriangledown}$	34
I-TFIDF-SIM	0.0140	21	0.0013$^{\triangledown}$	10	0.0654	17	**0.0800**$^{\blacktriangledown}$	34
% Change	-11.9%		+32.6%		-17.8%		-9.8%	

Similarity Fusion Table 4.5 shows the results of the experiments with the user-based and item-based filtering variants of the SimFuseCF algorithm. Fusing the different similarities together does not unequivocally produce better recommendations. For user-based filtering we see modest improvements of up to 26% for the two bookmark data sets BibBoo and Delicious. Similarity fusion, however, does not help us on the BibArt and CiteULike data sets. For item-based filtering, we see small improvements in MAP scores for two runs,

BibBoo and Delicious after fusing the item-based similarities from the two best component runs. The other two item-based SimFuseCF runs perform worse than the best component runs. However, none of the performance increases or almost none of the decreases are statistically significant. It is interesting to note that in two cases the SimFuseCF algorithm actually performs worse than both component runs. Apparently, the combined similarities cancel each other out to a certain extent there.

Finally, when we look at the optimal λ values we see that for user-based filtering the λ values are closer to 1, while for item-based filtering the λ values are closer to 0. This is to be expected as it corresponds to assigning more weight to the similarities of the best performing component run of each fusion pair.

Table 4.5: Results of the SimFuseCF algorithm. Reported are the MAP scores as well as the optimal number of neighbors N and the optimal value of λ. We report each of the best performing component runs and print the best-performing SimFuseCF runs in bold for both user-based and item-based filtering. The percentage difference between the best component run and the best fused run is indicated in the bottom rows of the two tables.

Runs	bookmarks						articles					
	BibSonomy			Delicious			BibSonomy			CiteULike		
	MAP	N	λ	MAP	N	λ	MAP	N	λ	MAP	N	λ
Best UB CF run	0.0277	13	-	0.0046	15	-	**0.0865**	4	-	**0.0757**	15	-
	(U-BIN-IDF-SIM)			(U-BIN-SIM)			(U-BIN-SIM)			(U-BIN-IDF-SIM)		
Best UB tag run	0.0102	5	-	0.0017	11	-	0.0459	6	-	0.0452	3	-
	(UT-BIN-SIM)			(UT-BIN-SIM)			(UT-JACCARD-SIM)			(UT-BIN-SIM)		
User-based fusion	**0.0350**	8	0.8	**0.0056**	45	0.7	0.0319$^{\triangledown}$	8	0.8	0.0554$^{\blacktriangledown}$	25	0.7
% Change	+26.4%			+21.7%			-63.1%			-26.8%		

Runs	bookmarks						articles					
	BibSonomy			Delicious			BibSonomy			CiteULike		
	MAP	N	λ	MAP	N	λ	MAP	N	λ	MAP	N	λ
Best IB CF run	0.0244	34	-	0.0027	25	-	0.0737	49	-	**0.0887**	30	-
	(I-BIN-IDF-SIM)			(I-BIN-SIM)			(I-BIN-IDF-SIM)			(I-BIN-IDF-SIM)		
Best IB tag run	**0.0370**	21	-	0.0101	23	-	0.1100	5	-	0.0814	34	-
	(IT-JACCARD-SIM)			(IT-BIN-SIM)			(IT-TFIDF-SIM)			(IT-DICE-SIM)		
Item-based fusion	0.0348	15	0.3	**0.0102**	25	0.3	**0.1210**	14	0.1	0.0791	28	0.4
% Change	-5.9%			+1.0%			+10.0%			-10.8%		

4.4.5 Discussion

Below we discuss the experimental results of our three algorithms based on tag overlap (TOBCF), tagging intensity (TIBCF), and similarity fusion (SimFuseCF).

Tag Overlap Similarity Earlier we saw that the benefits of using tag overlap are dependent on the type of TOBCF algorithm. While user-based TOBCF does not seem to get a boost from user-user similarities based on tag overlap, item-based TOBCF improves markedly by using tag overlap between items as its similarity metric.

Why do we see this performance dichotomy? Earlier, in Subsection 4.4.1, we put forward that the reduction in sparsity from using tag overlap could produce better recommendations for item-based filtering. On average, the number of tags assigned to an item is 2.5 times

higher than the number of users who have added the item. This means that, on average, item profile vectors from the **IT** matrix are less sparse than item profile vectors from the **UI** matrix. Using more values in the similarity calculation leads to a better estimate of the real similarity between two items. We believe this is why using tag overlap works better for item-based filtering. For user-based TOBCF this difference is not as well-pronounced: in some data sets users have more items than tags on average, and more tags than items in other data sets. This explains why we do not see the same performance increase for the user-based filtering runs based on tag overlap.

We also noted earlier that the performance on the bookmarks data sets shows the strongest reaction to using tag overlap: item-based TOBCF performs best on the bookmark data sets, while user-based TOBCF performs worst on them. The explanation for the lower performance of user-based TOBCF on bookmarks is that, as we showed earlier, the profiles of bookmark users are more diverse topically speaking and therefore harder to match against other profiles. As the tags assigned by the users to their bookmarks are about as numerous as their items, this does not reduce the sparsity, thereby not providing any help in matching the users up correctly. For item-based TOBCF, in the bookmarks case the sparsity is reduced by a factor 3 on average, while in the articles case sparsity is only reduced by a factor of 1.5. This means that sparsity is reduced more for bookmarks, and subsequently that items can be matched better on tag overlap in the bookmarks case, thereby improving performance more than in the article case. In addition, performance on the bookmarks data sets was already lower, making it easier to achieve a bigger percentage-wise improvement. We do not have a specific explanation for the particularly large increase in performance of item-based TOBCF on the Delicious data set. However, it should be noted that even the best scores on Delicious are still considerably lower than on the other data sets; the size of the Delicious data set with its magnitude more items makes recommendation difficult, no matter what metrics or algorithms are used.

Tagging Intensity Similarity We found that the tagging intensity, i.e., the number of tags users assign to items, is not a good source of user and item similarity. Most of our TIBCF runs that used tagging intensity similarities performed considerably worse than the baseline k-NN algorithm that uses transaction data to locate the nearest neighbors. How can we explain this? The simplest and most likely explanation is the number of tags simply is not a good predictor for interest in an item, and that the topical overlap between items cannot be captured by the number of tags assigned to them.

One way of investigating this is by looking at the topical overlap between users and items as we did in Subsection 4.3.3 using the tags. We assume that, more often than not, a user will be interested in new items that share some overlap in topics with the items already owned by that user. If tagging intensity is indeed a good indicator of topical overlap, then the similarity between two users in tagging intensity should be a good predictor of the topical similarity between those users, and vice versa. We therefore correlate (1) the cosine similarity between two users based on the tags they assigned with (2) the cosine similarity between two users based on how intensively they tagged their items[7]. What we find is

[7]Effectively, this means we are correlating the user similarities of the UT-TF-SIM run with the user similarities of the U-TF-SIM run.

that there is a negligible correlation between these similarities: the correlations range from -0.017 to 0.076.

We posited earlier that the latent complexity of the items might govern how many tags are needed to describe them—more complex items might simply require more tags. We therefore also correlated the item similarities based on tag overlap with the item similarities based on how intensely they were tagged[8]. Here again, we found similar, weak correlations between the two similarity sets. These findings lead us to conclude that looking at tagging intensity is not a good way of locating nearest neighbors in social bookmarking systems. Finally, it is interesting to note that tagging intensity similarities did produce a small improvement in MAP scores on the user-based runs on Delicious. We do not have a clear explanation for this.

Similarity Fusion The results of our SimFuseCF algorithm were not clear-cut: some runs on some data sets saw an increase in performance, while others saw performance drops, sometimes even below the scores of the original component runs. User-based SimFuseCF worked best on the two bookmark data set BibBoo and Delicious. Here, the sum of the parts is greater than the whole as both usage-based information and tag overlap are best combined to match users. There appears to be no clear pattern in when similarity fusion yields improvements and when it does not. From the lack of statistically significant results we may conclude that similarity fusion is neither an effective nor an efficient method of recommendation: scores are not improved significantly, which does not warrant the double effort of calculating two sets of similarities and merging them—computationally the most expensive part of the k-NN algorithm. We believe this to be because the distributions of the two set of similarities are too different even after normalization. If the optimal number of neighbors is very different for the two algorithms using the two sets of similarities, fusing the similarities themselves does not results in the best of both worlds, but rather a sub-optimal compromise between the optimal neighborhood sizes.

4.5 Related work

Social bookmarking and social tagging in general are relatively new phenomena and as a result there is not a large body of related work. We saw earlier in Chapter 2 that the majority of the literature so far has focused on the information seeking and organization aspects of tags and on the benefits of tags for information retrieval and Web search. In terms of recommendation, most of the research efforts have focused on tag recommendation. With the exception of the 2008 Discovery Challenge, there is also an striking absence of large scale experimentation and evaluation initiatives such as TREC or CLEF in the field of IR. This lack of standardized data sets combined with the novelty of social bookmarking makes for a relatively small space of related work on item recommendation for social bookmarking. We discuss three different types of related work: graph-based, memory-based, and model-based recommendation for social bookmarking

[8] Effectively correlating the item similarities of IT-TF-SIM runs with item similarities of I-TF-SIM runs.

Graph-based Recommendation One of the first approaches to recommendation for social bookmarking websites was done by Hotho et al. (2006a), who proposed a graph-based algorithm called *FolkRank*. They start by constructing an undirected tripartite graph of all users, items, and tags, and perform the same aggregations over these three dimensions as we described in Section 4.1 to arrive at the 2D projections **UI**, **UT**, and **IT**. They combine these into a single square matrix with as many rows as nodes in the original tripartite graph. Like PageRank (Page et al., 1998), the FolkRank algorithm is based on a random walk model that calculates the fully converged state transition probabilities by taking a walk of infinite length. The probabilities then represent the rank-ordering of the users, items, and tags on popularity. FolkRank also allows for the incorporation of a preference vector, similar to the teleporting component of PageRank. In this preference vector, specific users, items, or tags can be assigned a higher weight to generate user- or topic-specific rankings. They empirically evaluate their algorithm on a self-crawled Delicious data set, making it difficult to compare with other approaches. Clements et al. (2008a) also proposed a random walk model for item recommendation. They test their approach on a data set based on LibraryThing, which includes both tagging and rating information. They construct a similar matrix like Hotho et al., but include the ratings matrix **R** from their data set instead of the **UI** matrix. They also incorporate self-transition probabilities in the matrix and use the walk length as an algorithm parameter. We describe this approach in more detail in the next section when we compare it directly with our own approaches.

Memory-based Recommendation Adaptations of memory-based algorithms that include information about the tags assigned by users to items have also been proposed. Szomszor et al. (2007) proposed ranking items by the tag overlap with the active user's tag cloud and compare it to popularity-based recommendation. They take on the task of movie recommendation based on the Netflix data set[9] and harvest the tags belonging to each movie from the IMDB[10]. Their approach corresponds to calculating the tag overlap on the regular **UT** matrix using $sim_{UT-cosine}$ and tf·idf weighting. They found that tag overlap outperformed popularity-based recommendation. Yin et al. (2007) also calculate direct user-item similarities in their approach to recommending scientific literature. Nakamoto et al. (2007) augmented a user-based k-NN algorithm with tag overlap. They calculate the similarities between users using cosine similarity between the user tag profiles (i.e., $sim_{UT-cosine}$ on the regular **UT** matrix with tag frequency weighting). They evaluated their approach in Nakamoto et al. (2008), where they compared it with popularity-based recommendation, which was outperformed by their tag-augmented approach. In their paper, Tso-Sutter et al. (2008) also propose a tag-aware k-NN algorithm for item recommendation. In calculating the user and item similarities they include the tags as additional items and users respectively. They then calculate cosine similarity on these extended profile vectors and fuse together the predictions of the user-based and item-based filtering runs. We describe this approach in more detail in the next section when we compare it directly with our own approaches. Zanardi and Capra (2008) propose an approach called Social Ranking for tag-based search in social bookmarking websites. Inspired by CF techniques, to find content relevant to the query tags, they first identify users with similar interests to the active users. Content tagged by those users is scored higher, commensurate with the similarity between users based on

[9] http://www.netflixprize.com/
[10] http://www.imdb.com/

cosine similarity of the tag profiles (i.e., $sim_{UT-cosine}$ on the regular **UT** matrix with tag frequency weighting). In addition, they expand the original query tags with related tags to improve recall. Tag similarity is calculated on the **TI** matrix using cosine similarity and item frequency weighting. Their algorithm showed promising performance on a CiteULike data set compared to popularity-based rankings of content. Finally, Amer-Yahia et al. (2008) explore the use of item overlap and tag overlap to serve live recommendations to users on the Delicious website. They focus especially on using the nearest neighbors for explaining the recommendations: why was a certain item or user recommended?

Model-based Recommendation Symeonidis et al. (2008b) were among the first to propose a model-based approach that incorporates tagging information. They proposed using tensor decomposition on the third-order folksonomy tensor. By performing higher-order SVD, they approximate weights for each user-item-tag triple in the data set, which can then be used to support any of the recommendation tasks. They evaluated both item and tag recommendation on a Last.FM data set (Symeonidis et al., 2008b,a). Comparing it to the FolkRank algorithm (Hotho et al., 2006a), they found that dimensionality reduction based on tensor decomposition outperforms the former approach. Wetzker et al. (2009) take a Probabilistic Latent Semantic Analysis (PLSA) approach, which assumes a latent lower dimensional topic model. They extend PLSA by estimating the topic model from both user-item occurrences as well as item-tag occurrences, and then linearly combine the output of the two models. They test their approach on a large crawl of Delicious, and find it significantly outperforms a popularity-based algorithm. They also show that model fusion yields superior recommendation independent of the number of latent factors.

4.6 Comparison to Related Work

While several different classes of recommendation algorithms are reported to have been modified successfully to include tag, it remains difficult to obtain an overview of best practices. Nearly every approach uses a different data set, crawled from a different social bookmarking website in a different timeframe. Looking closer, we can also find a large variation in the way these data sets are filtered on noise in terms of user, item, and tag thresholds, and the majority of approaches are filtered more strongly than we proposed in Subsection 3.4.1. There is also a lack of a common evaluation methodology, as many researchers construct and motivate their own evaluation metric. Finally, with the exception of Symeonidis et al. (2008b) who compared their approach with FolkRank, there have been no other comparisons of different recommendation algorithms on the same data sets using the same metric, making it difficult to draw any definite conclusions about the algorithms proposed.

In this thesis, we attempt to alleviate some of these possible criticisms. With regard to the data sets, we have enabled the verification of our results by selecting publicly available data sets. In addition, we follow the recommendations of Herlocker et al. (2004) in selecting the proper evaluation metric. In this section, we will compare two of the state-of-the-art graph-based CF (GBCF) approaches with our own TBCF approaches on our data sets with the same metrics to properly compare the different algorithms.

In Subsections 4.6.1 and 4.6.2 we describe two GBCF algorithms in more detail. We present the experimental results of the GBCF algorithms in Subsection 4.6.3, and contrast them with the results of our best-performing TBCF algorithms. We discuss this comparison in Subsection 4.6.4.

4.6.1 Tag-aware Fusion of Collaborative Filtering Algorithms

The first GBCF approach to which we want to compare our work is that of Tso-Sutter et al. (2008). In their paper, they propose a tag-aware version of the standard k-NN algorithm for item recommendation on social bookmarking websites. We elect to compare our work to this algorithm, because it bears many similarities, yet calculates the user and item similarities in a different manner. Their approach consists of two steps: (1) *similarity calculation* and (2) *similarity fusion*. In the first step, they calculate the similarities between users and between items based on the **R** matrix, but extend this user-item matrix by including user tags as items and item tags as users. Effectively, this means they concatenate a user's profile vector $\overrightarrow{u_k}$ with that user's tag vector $\overrightarrow{d_k}$, which is taken from $\mathbf{UT}_{binary}$. For item-based filtering the item profile vector $\overrightarrow{i_l}$ is extended with the tag vector for that item $\overrightarrow{f_l}$, also taken from $\mathbf{IT}_{binary}$. Figure 4.4 illustrates this process.

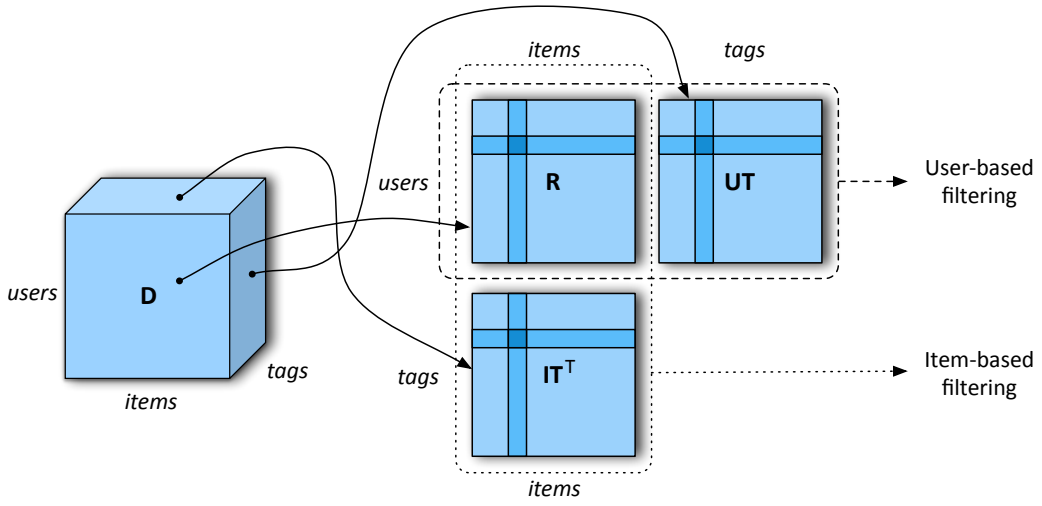


Figure 4.4: Extending the user-item matrix for tag-aware fusion. For user-based filtering, the **UT** matrix is appended to the normal **UI** matrix so that the tags serve as extra items to use in calculating user-user similarity. It does so by including user tags as items and item tags as users. For item-based filtering, the transposed **IT** matrix is appended to the normal **UI** matrix so that the tags serve as extra users to use in calculating item-item similarity. Adapted from Tso-Sutter et al. (2008).

By extending the user and item profile vectors with tags, sparsity is reduced when calculating the user or item similarities, compared to using only transaction data from **R** to calculate the similarities. Adding the tags also reinforces the transaction information that is already present in $\overrightarrow{u_k}$ and $\overrightarrow{i_l}$. At the end of this phase they use the k-NN algorithm with cosine similarity to generate recommendations using both user-based and item-based filtering. When generating recommendations, the tags are removed from the recommendation lists; only

items are ever recommended. While this is related to our idea of similarity fusion, it is not completely the same. We fuse together the similarities calculated on the separate $\overrightarrow{u_k}$ and $\overrightarrow{d_k}$ vectors in the case of, for instance, user-based filtering[11]. Tso-Sutter et al. first fuse together the $\overrightarrow{u_k}$ and $\overrightarrow{d_k}$ vectors and then calculate the similarity between profile vector pairs. The difference is identical for item-based filtering.

In the second phase of their approach, similarity fusion, Tso-Sutter et al. (2008) fuse the predictions of the user-based and item-based filtering algorithms together, to try to effectively capture the 3D correlations between users, items, and tags in social bookmarking data sets. Their fusion approach was inspired by Wang et al. (2006a), who proposed two types of combinations: (1) fusing user- and item-based predictions, and (2) using the similar item ratings generated by similar users. Tso-Sutter et al. (2008) only considered the first type of combinations as the second type did not provide better recommendations for them. We also employed this restriction. They fused the user- and item-based predictions by calculating a weighted sum of the separate predictions. In our comparison experiments, we evaluated both the fused predictions as well as the separate user-based and item-based filtering runs using the extended similarities. We refer to the latter two runs as U-TSO-SUTTER-SIM and I-TSO-SUTTER-SIM. The optimal combination weights were determined using our 10-fold cross-validation setup. Tso-Sutter et al. (2008) tested their approach on a self-crawled data set from Last.FM against a baseline k-NN algorithm based on usage similarity. They reported that in their experiments they found no improvements in performance using these extended similarities in their separate user-based and item-based runs. They did report significant improvements of their fused approach over their baseline runs, showing that their fusion method is able to capture the 3D relationship between users, items, and tags effectively. We refer the reader to Tso-Sutter et al. (2008) for more details about their work.

4.6.2 A Random Walk on the Social Graph

The second GBCF approach against which we wish to compare our work is the random walk method of Clements et al. (2008a). They propose using a personalized Markov random walk on the tripartite graph present in social bookmarking websites. While others have used random walks for recommendation in the past (Aggarwal et al., 1999; Yildirim and Krishnamoorthy, 2008; Baluja et al., 2008), applying them to the tripartite social graph is new. Furthermore, the model allows for the execution of many different recommendation tasks, such as recommending related users, interesting tags, or similar items using the same elegant model. Clements et al. (2008a) represent the tripartite graph of user, items, and tags, created by all transactions and tagging actions, as a transition matrix **A**. A random walk is a stochastic process where the initial condition is known and the next state is given by a certain probability distribution. **A** contains the state transition probabilities from each state to the other. A random walk over this social graph is then used to generate a relevance ranking of the items in the system. The initial state of the walk is represented in the *initial state vector* $\overrightarrow{v}_0$. Multiplying the state vector with the transition matrix gives us the transition probabilities after one step; multi-step probabilities are calculated by repeating $\overrightarrow{v}_{n+1} = \overrightarrow{v}_n \cdot \mathbf{A}$ for the desired walk length n. The number of steps taken determines

[11] I.e. the similarities from the U-BIN-SIM and UT-BIN-SIM runs.

the influence of the initial state vector versus the background distribution: a longer walk increases the influence of **A**. A walk of infinite length ($\overrightarrow{v}_\infty$) results in the steady state distribution of the social graph, which reflects the background probability of all nodes in the graph. This is similar to the PageRank model (Page et al., 1998) for Web search and similar, but not identical, to the FolkRank algorithm (Hotho et al., 2006a). The transition matrix **A** is created by combining the usage and tagging information present in the **R**, **UT**, and **IT** matrices into a single matrix. In addition, Clements et al. (2008a) include the possibility of self-transitions, which allows the walk to stay in place with probability $\alpha \in [0,1]$. Figure 4.5 illustrates how the transition matrix **A** is constructed.

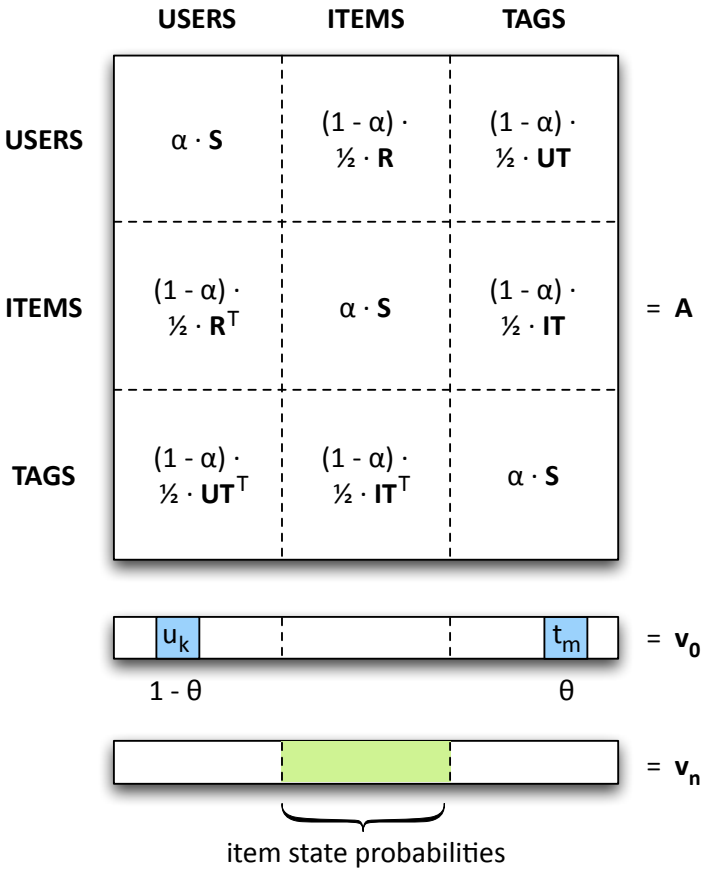


Figure 4.5: The transition matrix **A** is constructed by combining the **R**, **UT**, and **IT** matrices and their transposed versions. Self-transitions are incorporated by super-imposing a diagonal matrix of ones **S** on the transition matrix, multiplied by the self-transition parameter α. In the initial state vector the θ parameter controls the amount of personalization for the active user. The highlighted part of the final probability vector $\overrightarrow{v}_n$ after n steps are the item state probabilities; these are the final probabilities that we rank the items on for the active users.

A is a row-stochastic matrix, i.e., all rows of **A** are normalized to 1. Clements et al. (2008a) introduce a third model parameter θ that controls the amount of personalization of the random walk. In their experiments with personalized search, two starting points are assigned in the initial state vector: one selecting the user u_k and one selecting the tag t_m they wish to retrieve items for. The θ parameter determines the influence of the personal profile versus this query tag. In our case, we are only interested in item recommendation

based on the entire user profile, so we do not include any tags in the initial state vector. This corresponds to setting θ to 0 in Clements et al. (2008a). In addition, we set α, the self-transition probability, to 0.8 as recommended by Clements et al.. We optimize the walk length n using our 10-fold cross-validation setup. After n steps, the item ranking is produced by taking item transition probabilities from $\overrightarrow{v}_n$ for the active user ($\overrightarrow{v}_n(K+1,\ldots K+L)$) and rank-ordering them by probability after removal of the items already owned by the active user. An advantage of the random walk model of Clements et al. (2008a) is that it can support many different recommendation tasks without changing the algorithm. Supporting tag recommendation instead of item recommendation, for instance, can be done by simply selecting a user and an item in the initial state vector, and then ranking the tags by their probabilities from the final result vector. Item recommendation for entire user groups could also be supported by simply selecting the group users in the initial state vector, and then ranking by item probabilities.

Clements et al. (2008a) tested their approach on a data set based on LibraryThing[12], which includes both tagging and rating information. In addition, they artificially constructed a narrow version of this LibraryThing folksonomy to compare the effectiveness of their method on collaborative and individual tagging systems. They compared different parameter settings of their random walk algorithm and found that, because of the lower density of the narrow folksonomy, it is difficult to retrieve and recommend items in an individual tagging system. We refer the reader to Clements et al. (2008a) for more details about their work.

4.6.3 Results

Table 4.6 shows the results of the tag-aware fusion and random walk algorithms on our four data sets. The best-performing CF runs are also listed in the table. What we see, is that of the two runs using the extended user-item matrices for similarity comparison, I-TSO-SUTTER-SIM performs better than U-TSO-SUTTER-SIM. While U-TSO-SUTTER-SIM performs better than using simple usage-based similarities from Section 4.3 on three of the data sets, it does not improve upon our best CF runs. The I-TSO-SUTTER-SIM, however, does: the extended item similarities outperform the standard tag overlap similarities on every data set. The performance increases there range from 16% to 37%.

Fusing the user- and item-based predictions produces the best results so far. For all four data sets, the fused predictions improve performance by 18% to 64%, depending on the data set. Performance on CiteULike is improved significantly. For most of the data sets, the optimal λ assigns more weight to the item-based predictions, which also yield the best results on their own. The differences between the fused run and the I-TSO-SUTTER-SIM run are not statistically significant.

The random walk model does not fare as well on any of our four data sets: performance is lower than our best CF run and it is especially bad on the Delicious data set. The random walk method does outperform the popularity-based baseline on the BibBoo, BibArt, and CiteULike data sets.

[12] http://www.librarything.com/

Table 4.6: Results of the tag-aware and random walk approaches. Reported are the MAP scores as well as the optimal number of neighbors N. For the random walk model, N corresponds to the walk length n. In addition, we report the best-performing CF runs using usage-based similarity and tag overlap similarity. The best-performing runs for each data set are printed in bold. The percentage difference between the best approaches from related work and our best CF runs is indicated in the bottom rows of the two tables.

Runs	bookmarks				articles			
	BibSonomy		Delicious		BibSonomy		CiteULike	
	MAP	N	MAP	N	MAP	N	MAP	N
Best CF run	0.0370	3	0.0101	23	0.1100	7	0.0887	30
	(IT-JACCARD-SIM)		(IT-BIN-SIM)		(IT-TFIDF-SIM)		(I-BIN-IDF-SIM)	
U-TSO-SUTTER-SIM	0.0303	13	0.0057	54	0.0829	13	$0.0739^{\triangledown}$	14
I-TSO-SUTTER-SIM	0.0468	7	0.0125	13	0.1280	11	$0.1212^{\blacktriangle}$	10
Tag-aware fusion	**0.0474**	$\lambda = 0.5$	**0.0166**	$\lambda = 0.3$	**0.1297**	$\lambda = 0.2$	$\mathbf{0.1268}^{\blacktriangle}$	$\lambda = 0.2$
Random walk model	0.0182	5	$0.0003^{\blacktriangledown}$	3	0.0608	8	$0.0536^{\blacktriangledown}$	14
% Change	+28.1%		+64.4%		+17.9%		+43.0%	

4.6.4 Discussion

According to our results on the four data sets, extending the user-item matrix by including user tags as items and item tags as users is the superior way of calculating user and item similarities. Both for user-based and item-based filtering, combining usage and tag information before computing the similarities outperforms using just one of those similarity sources. We believe sparsity reduction to be a main reason for this. As user profile vectors increase in length, they also gain more non-zero elements. The density of the user profile vectors increases by 54% for the Delicious data set and by approximately 6% for the other data sets. This means that, on average, larger parts of the vectors can be matched against each other in the user similarity calculation, leading to more accurate user similarities. For item-based filtering, we see the same thing happening, leading to improved performance over the separate CF or TOBCF runs. As for the item-based runs with tag overlap, richer item descriptions lead to better matched items and therefore higher quality recommendations. These results are different from those reported by Tso-Sutter et al. (2008), who did not find CF with their extended similarities to outperform the standard CF algorithm. This effect might have been specific to their data set.

Fusing the user-based and item-based predictions leads to the best results, superior to any of our own CF-based runs. This is to be expected, as it combines the best of both worlds. We can confirm with our experiments that tag-aware fusion of CF algorithms is able to capture the three-dimensional correlations between users, items and tags. Although the tag-aware fusion method does not significantly improve over its component runs, it does provide consistent improvements. An interesting question is why our own proposed SimFuseCF algorithm did not show such consistent improvements. We believe this to be because the distributions of the two set of similarities were too different even after normalization. If the optimal number of neighbors is very different for the two algorithms using the two sets of similarities, fusing the similarities themselves does not result in the best of both worlds, but rather a sub-optimal compromise between the optimal neighborhood sizes. In Chapters 5 and 6 we will take another look at other possibilities for combining different recommendation techniques.

The random walk method is not competitive on our data sets with our best CF runs or with the tag-aware fusion method. We can think of several reasons for this. First, we performed more strict filtering our of data sets than Clements et al. (2008a) did, and the higher density of their LibraryThing data set could have led to better results. For instance, we removed only untagged posts from our crawls, whereas Clements et al. required all tags to occur at least five times. Increasing the density of our data sets through stricter filtering is likely to improve the results of all the approaches discussed so far. A second possible reason for the poor performance is the lack of explicit ratings. While Clements et al. have explicit, five-star ratings in their LibraryThing data set, we only have implicit transaction patterns to represent item quality. However, in their work they did not find any significant differences when using the ratings from the **UI** matrix. Finally, we did not perform extensive parameter optimization: we only optimized the walk length n, but not the self-transition probability α, which could have positively influenced performance on our data sets.

The random walk method performs much better on the article data sets than on the bookmark data sets. We believe this to be because the BibArt and CiteULike data sets comprise a specific domain of scientific articles. We reported earlier that the user profiles of BibArt and CiteULike users are more homogeneous than the bookmark users, whose profiles reflect a larger variety of topics. The larger number of tags per user on the bookmark data sets, mean that the random walk has more nodes it can visit. This means the transition probabilities are spread out over more nodes, making it more difficult to distinguish between good and bad items for recommendation. This effect is especially pronounced on the Delicious data set, as is evident in the significantly lower MAP score there. Narrower domains such as scientific articles, books, or movies lead to a more compact network and make it easier for the random walk model to find the best related content. The data set used by Clements et al. (2008a) has a similarly 'narrow' domain as our BibArt and CiteULike data sets, making it easier to find related content and generate recommendations.

4.7 Chapter Conclusions and Answer to RQ 1

Collaborative filtering algorithms typically base their recommendations on transaction patterns such as ratings or purchases. In a social bookmarking scenario, the broad folksonomy provides us with an extra layer of information in the form of tags. In this chapter we focused on answering our first research question.

> **RQ 1** How can we use the information represented by the folksonomy to support and improve recommendation performance?

We started our exploration by comparing a popularity-based algorithm with the user-based and item-based variants of the standard nearest-neighbor CF algorithm. The only information used here consisted of the user-item associations in the folksonomy, and these experiments showed that personalized recommendations are preferable over ranking items by their overall popularity.

We then extended both nearest-neighbor CF variants with different tag similarity metrics, based on either the overlap in tags (TOBCF) or the overlap in how intensely items were tagged (TIBCF). We found that the performance of item-based filtering can be improved by using the item similarities based on the overlap in the tags assigned to those items. The reason for this is reduced sparsity in the item profile vectors; something we did not find in user-based filtering, which as a result did not benefit from using tag similarity. We found that bookmark recommendation was affected more strongly than reference recommendation. Using tagging intensity as a source of user or item similarity did not produce good recommendations, because the amount of tags a user assigns to an item is not correlated with its perceived value. We also examined merging usage-based and tag-based similarities together to get the best of both worlds. However, the results of this SimFuseCF algorithm were inconclusive, probably due to the different distributions of the sets of similarities.

To promote repeatability and verifiability of our experiments, we used publicly available data sets, and we compared our algorithms with two state-of-the-art GBCF algorithms. The first algorithm, based on random walks over the social graph, performed worse than our best tag-based approaches. The second approach was a tag-aware k-NN algorithm which merged usage and tag data before the similarity calculation instead of after. By combining two different representations and combining the results of two different algorithms, this approach outperformed our TOBCF, TIBCF, and SimFuseCF algorithms. From these results we may conclude that tags can be used successfully to improve performance, but that usage data and tagging data have to be combined to achieve the best performance.

Chapter 5

Exploiting Metadata for Recommendation

Social bookmarking services offer their users the possibility to annotate the content of their items with metadata. The extent of this annotation is largely dependent on the domain and the items being annotated. Services such as Delicious typically allow users to add metadata such as titles and descriptions to their Web bookmarks. Social reference managers reflect the more complex creation and publication process of scientific papers; CiteULike for instance allows its users to add all kinds of citation-related metadata such as author information, publication venue, and abstracts. In this chapter we investigate the role that metadata can play in recommending interesting bookmarks or references as formulated in our RQ 2.

RQ 2 How can we use the item metadata available in social bookmarking systems to provide accurate recommendations to users?

This general research question gives rise to two more specific subquestions.

RQ 2a What type of metadata works best for item recommendation?

RQ 2b How does content-based filtering using metadata compare with folksonomic recommendations?

To answer these three questions, we propose and evaluate two different approaches to using item metadata. The standard method for using metadata is *content-based filtering*, which focuses on representing the content in a system and then learning a profile of the user's interests. Subsequently, the content representations are matched against the user profile to find the items that are most relevant to that user. In our content-based filtering approach we propose matching the metadata assigned by active users directly with the metadata of all unseen items in a single step. Another perspective on exploiting the metadata is to see it as yet another source for calculating user and item similarities in addition to the transaction

patterns from Chapter 4. We can then plug the newly calculated user and item similarities into a standard Collaborative Filtering (CF) algorithm. The resulting algorithm is a hybrid of CF and content-based filtering, and we refer to this perspective as *hybrid filtering*.

Chapter 5 is organized as follows. In Section 5.1 we start by describing the metadata available in our data sets. We categorize the different metadata fields according to the function they perform in describing the items in our data sets. Section 5.2 discusses our different approaches to exploiting metadata for recommendation in more detail. We compare the results of these different approaches in Section 5.3. Section 5.4 discusses the related work, and we conclude this chapter in Section 5.5.

5.1 Contextual Metadata in Social Bookmarking

The type of metadata used to annotate content in social bookmarking systems can vary greatly with the domain used, its conventions, and between individual social bookmarking systems. This variability is also reflected in our four data sets. The two data sets dealing with Web bookmarks contain only four different metadata fields, DESCRIPTION, TAGS, TITLE, and URL—whereas the two data sets that cover scientific articles reflect the variety inherent their the more complicated process of creation. The BibArt and CiteULike data sets offer 24 and 27 different metadata fields respectively. Table 5.1 lists the different metadata fields we were able to obtain for our four data sets (see also Section 3.3). In the experiments described in this chapter, we have included the tags assigned to an item as an additional metadata field. Since we expect that the different metadata fields vary in how beneficial they are to recommendation performance, we would like to determine which (sets of) metadata fields contribute most to improving performance. For structuring our research, we divide the metadata into two different groups: *item-intrinsic* metadata and *item-extrinsic* metadata.

Item-intrinsic metadata fields relate directly to the content of the item being annotated. Three of the most representative examples include TITLE, DESCRIPTION, or ABSTRACT, all of which partly describe the content of the item. Other fields, such as AUTHOR and JOURNAL can also be seen as intrinsic to the item and its creation. The intuition behind assigning metadata fields to the item-intrinsic category is that these fields can be used as stand-alone sources for recommending other content. For instance, given a certain paper from a user's profile, papers with similar abstracts, papers written by the same author, or papers published at the same workshop are likely to be relevant recommendations.

In contrast, item-extrinsic metadata fields—such as PAGES, MONTH, or ENTRYTYPE—cannot be used to directly generate appropriate recommendations. We define extrinsic metadata as administrative information that could play a supporting role in generating recommendation, but never serve as a stand-alone source. For example, that fact that a paper was published in the same month or allocated to the same pages in some proceedings does not mean that all other papers published in March or on pages 17 through 24 are relevant to the user as well. The exact categorization of metadata fields into these two categories is system-dependent and depends largely on the usage of the different fields. However, we believe that every

data set is likely to have metadata fields that fall into one of these two categories, and that classifying them as such is relatively easy, regardless of the domain.

Table 5.1 gives an overview of how we assigned the metadata fields available to us to the different classes. We will use this categorization to guide the experiments in the rest of this chapter, and define different subsets of our data based on this dichotomy. We will run experiments using only the combination of all intrinsic metadata fields, and perform separate runs for each of the intrinsic fields individually to investigate their contribution. In addition, we will run experiments with the combined set of all intrinsic and extrinsic metadata fields. Because of the supporting nature of the extrinsic metadata, we do neither investigate the stand-alone usage of this subset, nor do we examine the influence of the extrinsic metadata fields separately.

Table 5.1: Metadata categorization and distribution over the four data sets at the post, item, and user level. Numbers in the table indicate the percentage of posts, items, or users that have a value assigned for this field. The top half of the table lists the intrinsic metadata fields and the bottom half the extrinsic metadata fields.

Metadata field	bookmarks						articles					
	BibSonomy			Delicious			BibSonomy			CiteULike		
	Post	Item	User	Post	Item	User	Post	Item	User	Post	Item	User
All intrinsic fields	98.4	100	100	98.4	100	100	98.4	100	100	98.4	100	100
ABSTRACT	-	-	-	-	-	-	10.4	13.8	80.2	66.4	67.7	99.8
AUTHOR	-	-	-	-	-	-	95.3	96.8	100	97.5	98.9	100
BOOKTITLE	-	-	-	-	-	-	38.9	40.2	92.2	13.5	13.4	67.5
DESCRIPTION	28.8	48.8	90.1	14.2	19.8	79.9	48.4	80	73.1	-	-	-
EDITOR	-	-	-	-	-	-	17	19.6	90.4	1.3	1.4	33.4
JOURNAL	-	-	-	-	-	-	34.8	37	96.4	65.4	67.8	98.9
NOTE	-	-	-	-	-	-	2.7	3.9	62.9	-	-	-
SERIES	-	-	-	-	-	-	4.7	5.3	68.3	0.3	0.4	11.9
TAGS	98.4	100	100	98.4	100	100	98.4	100	100	98.4	100	100
TITLE	98.4	100	100	98.4	99.9	100	98.4	100	100	98.4	100	100
URL	97	98.2	100	98.4	100	100	25	29.8	91.6	95.9	97	100
All extrinsic fields	-	-	-	-	-	-	98.4	100	100	98.4	100	100
ADDRESS	-	-	-	-	-	-	25.5	29.9	94.6	38.2	37.2	94.2
CHAPTER	-	-	-	-	-	-	0.8	1.0	23.4	0.1	0.1	4.1
DAY	-	-	-	-	-	-	0.5	0.6	24.0	-	-	-
EDITION	-	-	-	-	-	-	1.5	1.9	43.7	0.1	0.1	6.2
ENTRYTYPE	-	-	-	-	-	-	100	100	100	100	100	100
HOWPUBLISHED	-	-	-	-	-	-	2.4	4.0	52.1	9.0	8.2	52.5
INSTITUTION	-	-	-	-	-	-	5.3	5.7	64.7	0.5	0.5	19.1
MONTH	-	-	-	-	-	-	15.4	17.2	90.4	62.6	60.5	99.5
NUMBER	-	-	-	-	-	-	20.7	21.9	93.4	55.6	56.4	98.6
ORGANIZATION	-	-	-	-	-	-	0.7	1.1	40.1	0.3	0.3	11.6
PAGES	-	-	-	-	-	-	54.9	57.5	98.8	67.4	68.5	98.7
PUBLISHER	-	-	-	-	-	-	41.6	44.4	97.6	32.7	31.1	95.6
SCHOOL	-	-	-	-	-	-	2.3	2.4	45.5	0.2	0.3	10.5
TYPE	-	-	-	-	-	-	1.4	1.6	40.1	0.0	0.0	0.4
VOLUME	-	-	-	-	-	-	36.6	38.5	97.0	60.6	61.9	98.9
YEAR	-	-	-	-	-	-	94.1	95.0	100	93.6	93.5	100

The numbers in the columns in Table 5.1 denote what percentage of the posts, items, and users have a value assigned to a particular metadata field. For instance, only 10.4% of the post in the BibArt data set were assigned an ABSTRACT. At the item level this increases to 13.8%, whereas 80.2% of all users have added at least one ABSTRACT to one of their posts. We report two observations here that give more insight into how users employ metadata to describe their content. We also conjecture how these observations could influence item recommendation based on metadata, and revisit these observations later.

First, going to a higher level of granularity typically decreases the sparsity. While a user might not assign a value to each metadata field for each of his posts, many users do so for at least one of their posts. The only two fields that have a value for every single post in all four data sets are TITLE and TAGS[1] It is interesting to note that only a minority of the posts have been assigned descriptions, but most users have done so for at least one of their items. The decrease in sparsity at higher levels of granularity level could imply that more fine-grained representations, such as a representation at the post level, could make the matching process more difficult because of missing data.

A second observation is that the overall usage rate of the different fields varies strongly between fields and data sets. Some of item-intrinsic fields are only very sparsely filled at the post and item level, such as SERIES, EDITOR, and BOOKTITLE, suggesting that recommendation based solely on these fields is likely to lead to a lack of accuracy and coverage. We see the same variation in sparsity for the extrinsic metadata fields, but this is not likely to lead to a lack of coverage since the extrinsic fields will only be used in combination with the intrinsic fields.

5.2 Exploiting Metadata for Item Recommendation

The presence of rich metadata offers many different possibilities for improving the recommendation performance. Metadata could, for instance, be used for re-ranking purposes or incorporated as priors. Certain metadata fields also offer the possibilities of exploring additional link graphs for recommendation, such as author or citation networks. In this chapter we focus on two specific methods of using the metadata to aid the recommendation process: *content-based filtering* and *hybrid filtering*. We discuss them below in Subsections 5.2.1 and 5.2.2 respectively. Both methods comprise a shared similarity matching step; we describe this similarity matching in more detail in Subsection 5.2.3. We conclude this section in Subsection 5.2.4 by describing which combinations of metadata fields we experiment with.

5.2.1 Content-based Filtering

The focus of content-based filtering for social bookmarking is on properly representing the content in our social bookmarking data sets. Based on these representations our aim is to construct an interest profile of an active user, and then use this profile to rank-order the unseen items by similarity to the profile, thereby approximating possible interest in those items. We propose two different algorithms for content-based filtering: *profile-centric matching* and *post-centric matching*. The difference between the two algorithms is the level of aggregation. We discuss both algorithms below and illustrate them by means of the same toy example for the sake of clarity.

Profile-centric Matching The *profile-centric matching* algorithm consists of two steps when generating recommendations for an active user: (1) generating collated user and item profiles, and (2) matching the active user's profiles against the item profiles of the unseen items.

[1]The latter is because we filtered out all untagged content, cf. Subsection 3.4.1.

In the first step, we start by collating all of the user's assigned metadata into a single user profile. The intuition here is that by aggregating all of the metadata assigned by the user we can completely capture his interests. Similarly, we construct item profiles that capture all of the metadata assigned to those items by all users in the training set.

The second step consists of matching the active user's profile against the item profiles on similarity to produce a ranking of all items. Items that have been assigned metadata that is similar to the metadata used by an active user to describe his items are likely to be good matches and thus good recommendations for that active user. After removing the items already in the active user's profile, we are left with the final rank-ordered list of recommendations. Figure 5.1 illustrates the profile-centric matching algorithm.

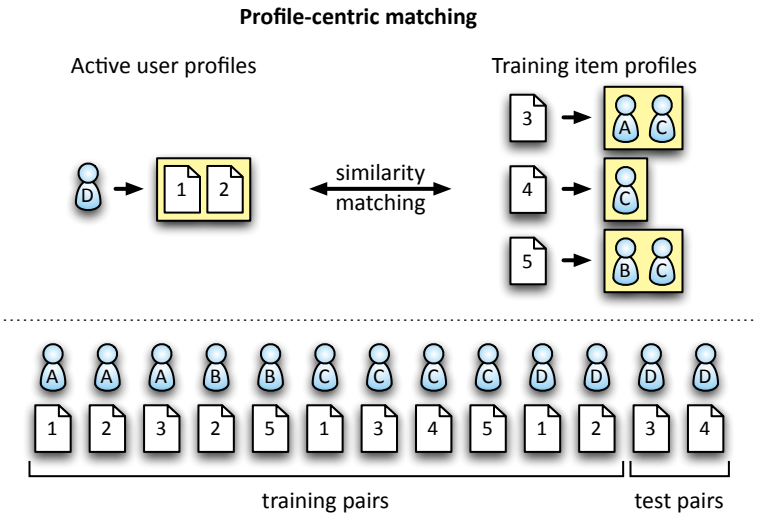


Figure 5.1: Visualization of our profile-centric matching algorithm for item recommendation. In the profile-centric approach, we directly match the active user's profile against the item profiles with the purpose of ranking the items in order of metadata similarity.

The bottom half of Figure 5.1 visualizes our toy data set with four users and five items. The user-item pairs in this toy data set have been divided into a training set and a test set. We have only one active user, D, for whom we are trying to predict items 3 and 4 as interesting. Using the profile-centric matching algorithm, we first build up a user profile for D that contains all of the metadata that D has added so far (items 1 and 2). Then the item profiles are built up for the items unseen by D, i.e., items 3, 4, and 5. The profile of D is then matched against item profiles 3, 4, and 5 to determine which of the items show the greatest textual match in metadata. We explain this similarity matching in more detail in Subsection 5.2.3.

A potential problem of the profile-centric matching algorithm is that by aggregating individual posts into high-level profiles we risk sacrificing precision for recall. Active users with a large number of items are likely to contain metadata for many different topics. With more and more different topics (and thus terminology) are covered by a profile, we can also expect a greater number of user (or item) profiles to partially match it. This is likely to increase recall, but could also reduce precision by producing false positives. At a certain

profile size nearly every other profile will match at least some small part of the active user's profile, making it harder to find the best neighbors for that user.

Post-centric Matching Our second content-based filtering algorithm, *post-centric matching*, operates at the level of individual posts instead of a higher level of aggregation. This could potentially reduce the influence of topical noise on the matching process when large quantities of metadata are aggregated[2]. The *post-centric matching* algorithm consists of three steps when generating recommendations for an active user. In the first step, we generate representations of all posts in the data set. We populate these representations with the metadata assigned to a specific post.

The second step then consists of matching each of the active user's posts separately against all the other posts of unseen items in the training set. This similarity matching is described in in more detail in Subsection 5.2.3. The second step leads to a list of matching posts in order of similarity for each of the active user's posts. Since the same item could have been added by multiple users, it is possible that the same item occurs multiple times in the ranked list of posts.

This leads us to the third step of post-centric matching: aggregating the ranked lists of posts into a single list of recommended items for the active user. To this end, we start by normalize the original similarity scores $sim_{original}$, as retrieval scores are not directly comparable between runs (Lee, 1997). We normalize $sim_{original}$ into $[0, 1]$ using the maximum and minimum similarity scores sim_{max} and sim_{min} using the following formula from Lee (1997):

$$sim_{norm} = \frac{sim_{original} - sim_{min}}{sim_{max} - sim_{min}}. \tag{5.1}$$

There are different methods for combining the different results lists into a single list of recommended items. Two examples include calculating the average score for each item or summing the different scores each item receives from each post it was retrieved from. Preliminary experiments with combination methods showed that a third method, calculating a rank-corrected sum of similarity scores for each item, gave the best performance. This means that if an item occurs many times it increases the odds of being recommended. At the same time, items that are recommended hundreds of times but with very low scores do not dominate the rankings unfairly. The final list of recommendations ranks all unseen items by their rank-corrected score $score(i)$ for an item i, calculated according to the following formula:

$$score(i) = \sum \frac{sim_{norm}(i)}{log(rank(i)) + 1}. \tag{5.2}$$

After removing the items already in the active user's profile, we are left with the final rank-ordered list of recommendations. Figure 5.2 illustrates the first two steps of the post-centric

[2]In contrast, having considerably less metadata when matching the representations could also result in more false negatives. Our evaluation will have to determine the effect of the different aggregation levels.

matching algorithm using the same toy data set as in Figure 5.2. First, the post representations are generated by the algorithm. Then, in the second step, each of user D's training posts is matched against all other, unseen posts. First, D's post of item 1 is matched against all other posts: user A's item 2 post, user A's item 3 post, user B's item 2 post, and so on. This results in a ranked lists of posts which serves as input to the third step of post-centric matching. The same post matching process then takes place for user D's post of item 2, again resulting in a ranked list of posts.

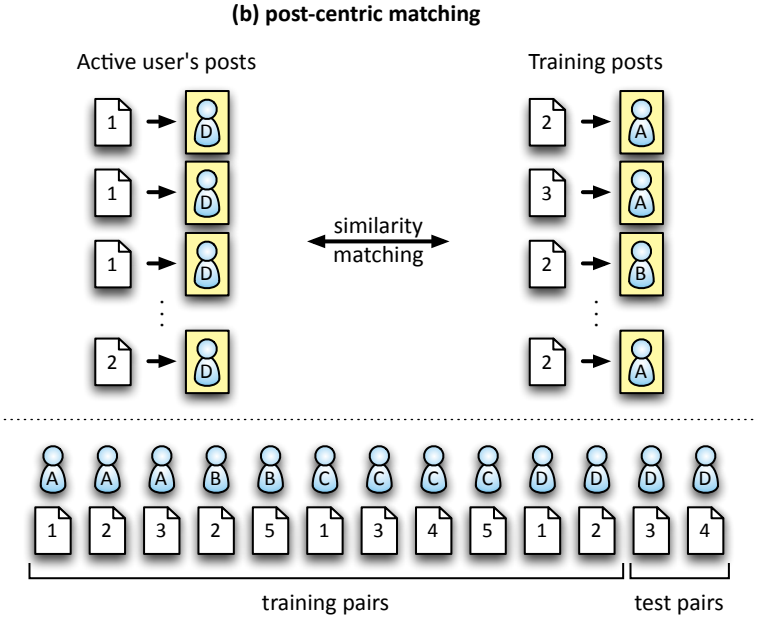


Figure 5.2: Visualization of our post-centric matching algorithm for item recommendation. In the post-centric approach, we match metadata at the post level with the purpose of locating the posts most similar to those of the active user in terms of metadata. The individual recommendations are then aggregated into a unified list.

5.2.2 Hybrid Filtering

In addition to focusing solely on using the metadata for recommendation, we also consider a hybrid approach that joins content-based filtering and CF. By combining these two techniques, we hope retain the best elements from both techniques. Combining the separate approaches can help diminish their individual shortcomings, and thus produce a more robust system. In our *hybrid filtering* approach we view metadata in social bookmarking systems as another source of information for locating the nearest neighbors in CF algorithms. Analogous to the CF algorithm described in Section 4.3, hybrid filtering also comes in two variants: (1) user-based hybrid filtering and (2) item-based hybrid filtering.

Instead of only looking at the overlap in items that two users (or two items) have in common when calculating user (or item) similarities, we can use the overlap in the metadata applied to items to determine the most similar neighbors. For the user-based hybrid filtering, for instance, users that describe their profile items using the same terminology are

likely to share the same interests, making them a good source of recommendations. This is similar to the way we used the tag clouds of users to calculate similarity between users in the user-based TOBCF algorithm in Chapter 4. In fact, TOBCF could also be considered to be a hybrid filtering approach, combining tag overlap with CF. In this subsection, we extend this approach to all metadata assigned to bookmarks or references. The user and item similarities we derive in this way are then plugged into the standard memory-based CF algorithms as described in Section 4.3. The resulting algorithm is a hybrid of CF and content-based filtering.

User-based hybrid filtering User-based hybrid filtering is the first variant of hybrid filtering. Analogous to the CF algorithm described in Section 4.3, it consists of two steps: (1) calculating the most similar neighbors of the active user, and (2) using those neighbors to predict item ratings for the active user. The second step is performed in the same manner as described earlier in Subsection 4.3.1.

As in the previous section, we approach the first step from an information retrieval perspective and calculate the textual similarities between users or items. For each user we generate user profile representations, and then retrieve the most relevant user profiles for all of the active users. These representations are constructed as follows. All of the metadata text of a user's posts is collated into a single user profile representation for that user. Figure 5.3 illustrates the first step in user-based hybrid filtering using the same toy data set from Subsection 5.2.1. In the first step, the user profiles are generated for all of the users. For user C, for instance, this means concatenating all the metadata assigned by C to items 1, 3, 4, and 5. After constructing the user profiles, the user similarities are then calculated as described in Subsection 5.2.3.

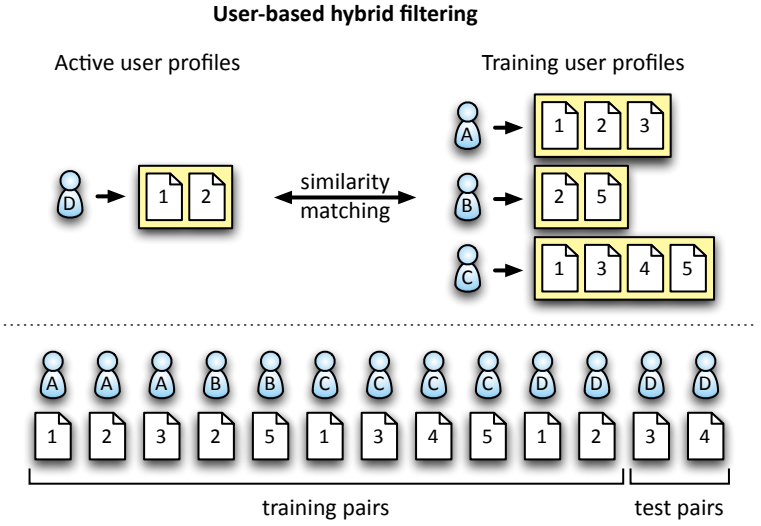


Figure 5.3: Visualization of our user-based hybrid filtering algorithm for item recommendation. In the user-based approach, the metadata assigned by a user to his items is concatenated to form a user profile. User similarities are then calculated by matching active user profiles with the other training user profiles.

Item-based hybrid filtering The second variant of hybrid filtering is item-based hybrid filtering, which again consists of two steps: (1) calculating the most similar neighbors of the active user's items, and (2) using those neighboring items to predict which unseen items the active user would. The second step is performed in the same manner as described earlier in Subsection 4.3.1. For each item we generate item profile representations, and then retrieve the most relevant item profiles for all of the active user's items. The item representations are constructed as follows. We create item profiles for each item by concatenating all of the metadata assigned to that item by all the users who have the item in their profile. This means that items are represented by their aggregated community metadata and not just by a single user's data. After constructing the item profiles, the item similarities are then calculated as described in Subsection 5.2.3.

Figure 5.4 illustrates the first step in item-based hybrid filtering using the same toy data set as before. In the first step, the item profiles are generated for all of the active user's items. For item 1 of user D, for instance, this means concatenating all the metadata assigned by users A, C, and D to item 1. After constructing the item profiles, the item similarities are then calculated as described in Subsection 5.2.3.

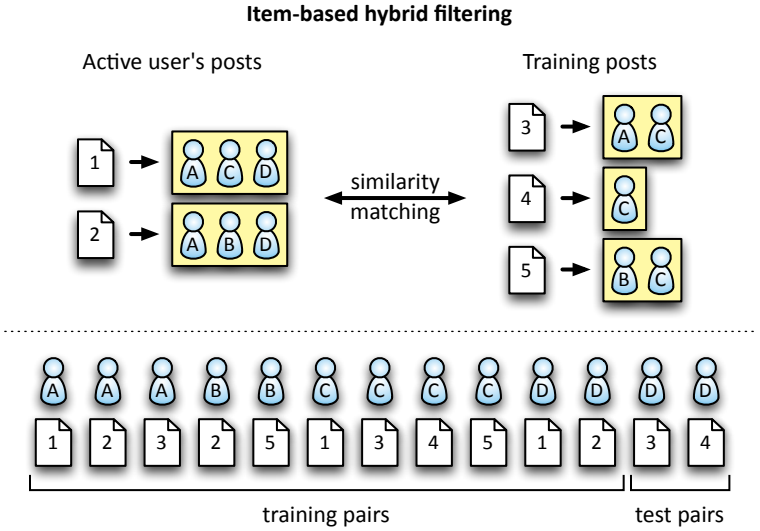


Figure 5.4: Visualization of our item-based hybrid filtering algorithm for item recommendation. In the item-based approach we combined the item metadata to form item profiles. The item similarities are then calculated by matching active item profiles with the other training item profiles.

5.2.3 Similarity Matching

Both the content-based and hybrid filtering algorithms contain a similarity matching step where user, item, or post representations are matched against each other on similarity in the metadata assigned. There are many different methods of calculating the similarity between representations. These methods are typically based on techniques from IR, machine learning, or both (see also Section 5.4). We chose to approach content-based and hybrid

filtering from an IR perspective because of the rich textual nature of many of the metadata fields.

To calculate the similarities between the different user, item, and post profiles, we used the open-source retrieval toolkit Lemur[3]. The Lemur toolkit implements different retrieval algorithms based on language modeling as well as probabilistic and Vector Space-based retrieval algorithms. We used version 2.7 of this toolkit to perform our experiments. The toolkit also offers support for options such as stemming and stop word filtering. We refer the reader to Strohman et al. (2005) for more information about the Lemur toolkit.

In order to decide which retrieval algorithm to use for our similarity matching, we compared four different algorithms (Strohman et al., 2005): (1) language modeling with Jelinek-Mercer smoothing, (2) language modeling with Dirichlet smoothing, (3) the OKAPI model, and (4) the Vector Space model with tf·idf term weighting. Preliminary experiments showed that language modeling with Jelinek-Mercer smoothing was the best-performing retrieval method. Jelinek-Mercer smoothing interpolates the language model of a user or item profile with the language model of a background corpus—in our case the training collection of all user or item profiles. In our experiments, the weighting parameter λ was set to 0.9. The language models we used were maximum likelihood estimates of the unigram occurrence probabilities. Our preliminary findings with regard to Jelinek-Mercer smoothing were in line with the findings of Zhai and Lafferty (2004), who found it to work better for verbose queries than other smoothing methods such as Dirichlet smoothing, and with our previous work on news article recommendation in Bogers and Van den Bosch (2007). We also examined stemming and stop word filtering using the SMART stop word list in our preliminary experiments. Based on the results we filtered stop words and did not perform stemming.

5.2.4 Selecting Metadata Fields for Recommendation Runs

In Section 5.1, we described the range of metadata fields we have available in our four data sets, and how we divided these into intrinsic and extrinsic metadata fields. We wish to address RQ 2a by examining which (combinations of) metadata fields provide the best performance for the four algorithms we described in the previous section. Experimenting with every single possible combination of metadata fields would lead to a combinatorial explosion of possible sets of metadata fields. In this section we describe which sets of metadata fields we decide to examine.

The item-intrinsic metadata fields relate directly to the content of the item being annotated. We will therefore examine the contribution of all intrinsic metadata fields separately, as well as running experiments using the set of all intrinsic metadata fields combined. In contrast, the item-extrinsic fields can only play a supporting role in generating recommendations. We will therefore only examine the extrinsic metadata fields in conjunction with all of the intrinsic metadata fields combined.

This results in the following experiments. For the two bookmarks data sets, BibBoo and Delicious, we have only four different intrinsic metadata fields and no extrinsic metadata. This

[3]Available at `http://www.lemurproject.org`

results in five different runs for both data sets: one run for each of the intrinsic metadata fields separately, and one run for all intrinsic metadata fields combined.

The data sets covering scientific articles, BibArt and CiteULike, contain more metadata fields. The BibArt data set offers 11 item-intrinsic metadata fields and 16 item-extrinsic metadata fields; CiteULike offers 9 item-intrinsic and 15 item-extrinsic metadata fields. This results in 13 different experimental runs on the BibArt data set: 11 separate runs for each of the intrinsic metadata fields, an additional run for all intrinsic metadata fields combined, and final run with all intrinsic and extrinsic metadata fields combined. For the CiteULike we have 11 experimental runs in total, because of two missing intrinsic metadata fields. This results in a total number of 34 experimental runs for each of the four algorithms, allowing us to determine what metadata works better for which algorithm.

5.3 Results

In this section we present the results of the four algorithms we described in Subsection 5.2.1 and 5.2.2 on the 34 experimental runs described in Subsection 5.2.4. For each algorithm we use the same tabular form to present our results: in the top part of the table we list the results of a particular algorithm using only the individual intrinsic metadata fields. The bottom part shows the results for the two runs with the combined metadata fields. We group together the results of the two content-based filtering algorithms in Subsection 5.3.1 and the two hybrid filtering algorithms in Subsection 5.3.2 for comparison purposes. In Subsection 5.3.3 we compare the results of recommendation using metadata to the best TBCF and GBCF algorithms from Chapter 4.

5.3.1 Content-based Filtering

Table 5.2 contains the results of the two content-based filtering approaches. When we compare the two algorithms, we see that, overall, the best-performing approach is the profile-centric approach where we match complete user profiles against complete item profiles. It outperforms the post-centric approach on three of our four data sets; significantly so on the CiteULike data set with an improvement of 117% ($p < 10^{-6}$). Only on the Delicious data set does post-centric matching perform significantly better ($p < 0.05$). If we look at all the individual runs, we see that on 22 of 34 runs the profile-centric approach achieves the best results. This advantage is even stronger on the article data sets, where 75% of the time the profile-centric approach performs best.

Metadata Comparison How do the individual fields contribute towards performance of the profile- and post-centric approaches? The best-performing single fields are `AUTHOR`, `DESCRIPTION`, `TAGS`, and `TITLE`, which provide the best individual results on all four data sets for both content-based filtering approaches. This is not surprising, as these fields are the least sparse of all the intrinsic fields, and they are aimed directly at describing the content of the items, more so than the conference or journal titles or the editors. In ten out of thirteen instances, these fields perform best in a profile-centric approach. On the CiteULike

Table 5.2: Results of the profile-centric (top) and post-centric (bottom) content-based filtering approaches. Reported are the MAP scores. The best-performing runs for each data set and approach are printed in bold.

Profile-centric approach

Runs	bookmarks		articles	
	BibSonomy	Delicious	BibSonomy	CiteULike
ABSTRACT	-	-	0.0132	0.0497
AUTHOR	-	-	0.0625	0.0422
BOOKTITLE	-	-	0.0135	0.0182
DESCRIPTION	0.0084	0.0004	0.0977	-
EDITOR	-	-	0.0063	0.0000
JOURNAL	-	-	0.0095	0.0121
NOTE	-	-	0.0167	-
SERIES	-	-	0.0108	0.0005
TAGS	0.0350	0.0013	0.0980	0.0593
TITLE	0.0138	**0.0014**	0.0502	0.0660
URL	0.0119	0.0012	0.0059	0.0165
All intrinsic fields	**0.0402**	0.0011	**0.1279**	0.0978
All intrinsic + extrinsic fields	-	-	0.1188	**0.0987**

Post-centric approach

Runs	bookmarks		articles	
	BibSonomy	Delicious	BibSonomy	CiteULike
ABSTRACT	-	-	0.0176	0.0254
AUTHOR	-	-	0.0347	0.0191
BOOKTITLE	-	-	0.0124	0.0079
DESCRIPTION	0.0110	0.0003	0.0809	-
EDITOR	-	-	0.0128	0.0000
JOURNAL	-	-	0.0045	0.0063
NOTE	-	-	0.0140	-
SERIES	-	-	0.0142	0.0011
TAGS	0.0159	**0.0036**	0.0715	0.0377
TITLE	0.0107	0.0018	0.0215	0.0285
URL	0.0146	0.0013	0.0071	0.0048
All intrinsic fields	**0.0259**	0.0023	**0.1190**	**0.0455**
All intrinsic + extrinsic fields	-	-	0.1149	0.0441

data set, these differences are all significant. Interestingly, on Delicious the best run is the post-centric run using only the TAGS field which performs better than all the other runs, and significantly so on all runs except the post-centric run with all intrinsic fields combined. Here, it appears that the extra information actually confuses the recommendation algorithm compared to using the individual fields.

In general though, the runs with all intrinsic fields combined outperform the runs using individual fields. On the article data sets, these differences are significant in the majority of the cases; on the bookmark data sets the intrinsic combined runs were only significantly better in a few cases. If we look at the runs where the two groups of fields are combined, we see that using only the intrinsic results in slightly better results than using both the extrinsic and the intrinsic metadata fields. In three out of four cases on the BibArt and CiteULike data sets, the runs with intrinsic fields are slightly better than the runs with all fields, but these differences are never significant. This suggests that adding the extrinsic fields that

describe non-content-based aspects of the items might confuse the k-NN algorithm slightly when making predictions.

5.3.2 Hybrid Filtering

Table 5.3 contains the results of the two hybrid filtering approaches: user-based and item-based hybrid filtering.

Table 5.3: Results of the user-based (top) and item-based (bottom) hybrid filtering approaches. Reported are the MAP scores as well as the optimal number of neighbors N. The best-performing runs for each data set and approach are printed in bold.

User-based approach

Runs	bookmarks				articles			
	BibSonomy		Delicious		BibSonomy		CiteULike	
	MAP	N	MAP	N	MAP	N	MAP	N
ABSTRACT	-	-	-	-	0.0121	2	0.0494	2
AUTHOR	-	-	-	-	0.0360	2	0.0605	2
BOOKTITLE	-	-	-	-	0.0274	2	0.0434	3
DESCRIPTION	0.0099	2	0.0007	8	0.0128	2	-	-
EDITOR	-	-	-	-	0.0066	2	0.0358	2
JOURNAL	-	-	-	-	0.0319	2	0.0409	2
NOTE	-	-	-	-	0.0203	2	-	-
SERIES	-	-	-	-	0.0231	2	0.0079	2
TAGS	0.0168	2	0.0016	13	0.0158	2	0.0405	4
TITLE	0.0183	2	0.0020	8	**0.0410**	2	**0.0608**	2
URL	**0.0218**	2	0.0029	13	0.0073	2	0.0536	2
All intrinsic fields	0.0197	2	**0.0039**	13	0.0155	2	0.0536	2
All intrinsic + extrinsic fields	-	-	-	-	0.0105	2	0.0545	2

Item-based approach

Runs	bookmarks				articles			
	BibSonomy		Delicious		BibSonomy		CiteULike	
	MAP	N	MAP	N	MAP	N	MAP	N
ABSTRACT	-	-	-	-	0.0179	17	0.0423	54
AUTHOR	-	-	-	-	0.0705	10	0.0325	9
BOOKTITLE	-	-	-	-	0.0104	5	0.0103	6
DESCRIPTION	0.0083	2	0.0003	8	0.0754	13	-	-
EDITOR	-	-	-	-	0.0081	7	0.0000	8
JOURNAL	-	-	-	-	0.0165	6	0.0096	11
NOTE	-	-	-	-	0.0165	2	-	-
SERIES	-	-	-	-	0.0094	7	0.0022	7
TAGS	**0.0399**	11	0.0014	4	0.0773	13	**0.0746**	21
TITLE	0.0059	8	0.0007	9	0.0300	23	0.0475	25
URL	0.0106	2	0.0006	3	0.0107	3	0.0078	3
All intrinsic fields	0.0267	14	**0.0017**	7	**0.1510**	21	0.0719	49
All intrinsic + extrinsic fields	-	-	-	-	0.1482	28	0.0716	49

We see that the best-performing runs for each data set are all from the item-based hybrid filtering approach. On the BibArt and Delicious data sets these differences are statistically significant and especially large at 268% ($p < 0.05$) and 112% respectively ($p < 0.01$). It is interesting to note that on the majority of the 28 individual runs, the user-based approach

performs best in 19 cases. However, most of these runs are using fields that were only sparsely filled. It is to be expected then that user-based filtering, the method that aggregates the most information, can best deal with these more sparsely populated fields.

Metadata Comparison How do the individual fields contribute towards performance of the user- and item-based approaches? As with the content-based filtering approaches, the best-performing single fields are AUTHOR, DESCRIPTION, TAGS, and TITLE, which provide the best individual results on all four data sets in 75% of the cases. In addition, however, we see that the URL, JOURNAL, and BOOKTITLE also tend to perform well, especially for the user-based approach. On the bookmark data sets BibBoo and Delicious, the runs using only the content of the URL actually perform best of all individual runs, which is surprising. Another interesting observation is that the TITLE field serves as a better source of user and item similarity on the article data sets than on the bookmark data sets. Apparently, the titles assigned to bookmarks are not as descriptive as the titles assigned to scientific articles, leading to this big performance gap.

If we look at the runs where the two different groups of fields are combined, we see again that there are only small performance differences between these two runs. In three out of four cases, the runs with intrinsic fields are slightly better, but these differences are never significant. Finally, we see that in five out of eight cases, the runs where all intrinsic fields are combined are actually outperformed by runs using only the metadata from a single field. This suggests that for calculating user and item similarities combining all the intrinsic fields often confuses the recommendation algorithms.

5.3.3 Comparison to Folksonomic Recommendation

In this subsection, we compare the best recommendation results obtained by using metadata with the best results obtained in Chapter 4 where we only used the folksonomy to generate recommendations. Table 5.4 shows the results of the best content-based and hybrid filtering runs, as well as the best CF and TOBCF runs from Chapter 4. In addition, we compare it to the tag-aware fusion approach of Tso-Sutter et al. (2008), the best-performing GBCF algorithm.

We can see in Table 5.4 that on three out of four data sets our recommendation algorithm that uses metadata is better than the best CF run. Only on the Delicious data set do all metadata-based approaches perform significantly worse than the CF runs that use only information from the folksonomy. The best-performing runs are all runs that use the combination of intrinsic and/or extrinsic fields to calculate the similarities between users and items. The best metadata-based approach is strongly dependent on the data set, but the profile-centric approach performs best on two of our four data sets, BibBoo and CiteULike. When we compare all metadata-based approaches with each other, we see that most differences are not statistically significant. On the BibArt data set, the item-based hybrid filtering approach is significantly better than the user-based approach ($p < 0.05$). On the CiteULike data set, the profile-centric approach also significantly outperforms the post-centric and user-based approaches.

Table 5.4: Results comparison of the metadata-based runs with our best folksonomic recommendation runs. Reported are the MAP scores as well as the optimal number of neighbors N. The best-performing metadata-based runs are printed in bold. In addition, we compare our best runs so far from both chapters with the tag-aware fusion algorithm we examined in Subsection 4.6.1. The percentage difference between our best CF approaches and between the tag-aware fusion method are listed in the appropriate rows.

Runs	bookmarks				articles			
	BibSonomy		Delicious		BibSonomy		CiteULike	
	MAP	N	MAP	N	MAP	N	MAP	N
Best CF run	0.0370	3	0.0101	23	0.1100	7	0.0887	30
	(IT-JACCARD-SIM)		(IT-BIN-SIM)		(IT-TFIDF-SIM)		(I-BIN-IDF-SIM)	
Profile-centric filtering	**0.0402**	-	0.0014$^{\blacktriangledown}$	-	0.1279	-	**0.0987**	-
	(all intrinsic)		(TITLE)		(all intrinsic)		(all extrinsic)	
Post-centric filtering	0.0259	-	0.0036$^{\triangledown}$	-	0.1190	-	0.0455$^{\blacktriangledown}$	-
	(all intrinsic)		(TAGS)		(all intrinsic)		(all extrinsic)	
User-based hybrid filtering	0.0218	2	**0.0039**$^{\triangledown}$	13	0.0410	2	0.0608$^{\blacktriangledown}$	2
	(URL)		(all intrinsic)		(TITLE)		(TITLE)	
Item-based hybrid filtering	0.0399	11	0.0017$^{\blacktriangledown}$	8	**0.1510**	21	0.0746	21
	(TAGS)		(all intrinsic)		(all intrinsic)		(TAGS)	
% Change over best CF run	+8.6%		-61.4%		+37.3%		+11.3%	
Tag-aware fusion	0.0474	-	0.0166	-	0.1297	-	0.1268	-
% Change over tag-aware fusion run	-15.1%		-50.0%		+16.4%		-22.2%	

If we compare the metadata-based runs to the tag-aware fusion algorithm, we find that tag-aware fusion is the strongest recommendation algorithm for social bookmarking we have seen so far. Compared to the metadata runs, only the item-based hybrid filtering approach on the BibArt data set outperforms the tag-aware fusion approach by 16% and this is not statistically significant. On the other three data sets, tag-aware fusion is the best approach. On the two largest data set Delicious and CiteULike these differences are even statistically significant ($p < 0.05$).

5.4 Related Work

We start in Subsection 5.4.1 by discussing the related work on content-based filtering. In Subsection 5.4.2 we then discuss the related work on hybrid filtering approaches in recommendation.

5.4.1 Content-based Filtering

While a significant amount of research has focused on CF for recommending interesting items, there has also been considerable work on content-based filtering, which can be seen as an extension of the work done on information filtering (Belkin and Croft, 1992). Typically, content-based filtering approaches focus on building some kind of representation of the content in a system and then learning a profile of the user's interests. The content representations are then matched against the user's profile to find the items that are most relevant to that user. The type of representations that are constructed depend on the learning methods employed, which typically include techniques from IR, machine learning, or

both. Content-based filtering has been applied to many different domains. We discuss approaches that cover six of these domains: news, movies, books, music, cultural heritage, and personal agents.

News Early work on content-based filtering in the news domain included the NewsWeeder system by Lang (1995), which used the words contained in newsgroup messages as its features and found that a probabilistic approach based on the Minimum Description Length principle outperformed tf·idf weighting. In their approach to personalization of an online newspaper, Maidel et al. (2008) used an ontology of news concepts to recommend interesting news articles to users, because the dynamic nature of news precludes the use of CF. Articles were rank-ordered based on the distance between ontological concepts covered by the user and item profiles. Their experiments showed that using a hierarchical ontology produces better recommendations compared to a flat ontology.

Movies Alspector et al. (1997) compared a CF approach to movie recommendation with content-based filtering. For their content-based component they built metadata representations of all movies using fields such as directory, genre, and awards, and used linear regression and classification and regression trees to learn user profiles and rank-order the items for those users. They found that CF performed significantly better than the content-based methods, but noted that this was likely due to the poor feature set they used.

Books Mooney and Roy (2000) describe LIBRA, a content-based book recommender system. In their approach, they crawled the book metadata from the Amazon website and represented each book as a bag-of-words vector. They then used a Naive Bayes classifier to learn user profiles and to rank-order unseen books for the user, with good results.

Music Music retrieval and recommendation is a fourth domain that content-based filtering has been applied to. It is one of the typical fields where content analysis of the audio signal is often difficult and costly, so metadata and tags are often used for content-based music retrieval instead. Whitman and Lawrence (2002), for instance, crawled textual descriptions of artists on web pages, online groups, and discussion forums to do content-based music recommendation. They then used this data to calculate artist similarity and recommend related artists. They calculated tf·idf-based similarity based on extracted noun phrases and adjectives, and attained good results compared to human-edited lists.

Cultural heritage De Gemmis et al. (2008) tackled the problem of cultural heritage personalization using content-based recommendation. They represented user profiles as feature vectors containing both the metadata and the tags assigned to a user's selected content, and used the Naive Bayes algorithm to learn user profiles and predict what unseen items might be of interest to the users. On their small data set, they found that using metadata and tags together results in the best performance.

Personal agents A sixth domain in which content-based filtering techniques are often used is that of personal agents, the so-called *Information Management Assistants* (IMAs), that locate and recommend relevant information for the user by inferring a model of the user's interests. We already discussed IMAs extensively earlier on in Subsection 2.1.4. We

noted that the majority of these IMAs approach content-based filtering from an IR perspective. Typically, they condense the Web pages or documents that a user has shown an implicit or explicit interest in, into a set of keywords that then sent off to various search engines. The results are subsequently ranked and presented to the users. The keywords are also commonly used to build up a representation of the user to further personalize the recommendations. This type of approach to content-based filtering inspired our own approaches, with the difference that we do not condense our item representations into queries, but instead let the retrieval engine take care of this.

5.4.2 Hybrid Filtering

We are not the first to propose a union of CF and content-based filtering. The advantages of both approaches are largely complementary. CF is the more mature of the two approaches and works best in a situation with a stable set of items and a dense user base that is divided into different neighborhoods. In such situations, they are good at exploring new topics and taking quality into account in the form of ratings. However, CF algorithms are known to have problems with users with unique tastes and so-called 'gray sheep' users with unremarkable tastes that are hard to categorize (Claypool et al., 1999). Content-based filtering methods are better at dealing with sparse, dynamic domains such as news filtering, and are better at recommending for users that are hard to peg. They are also known to produce focused, high precision recommendations, and to suffer less from the cold-start problem.

Several hybrid methods that try to combine the best of both worlds have been proposed over the years. We briefly describe five of them. Basu et al. (1998) were among the first to propose a hybrid recommender system that used both collaborative and content features to represent the users and items. The collaborative features captured what movies a user likes and the content features include metadata fields such as actors, directors, genre, titles, and tag lines. They used Ripper, a rule-based machine learning algorithm to learn user profiles and predict which items are interesting. They found that the combination of collaborative and content-based features did indeed produce the best results. Claypool et al. (1999) present a weighted hybrid recommender system that calculates a weighted average of the output of two separate CF and content-based filtering components. The CF component receives a stronger weight as the data sets grows denser, gradually phasing out the influence of the content-based component. They did not find any significant differences between the performance of the separate components or the combined version. A third hybrid filtering method is that by Baudisch (1999), who proposed an innovative approach to incorporating metadata into CF algorithms by joining the metadata descriptions to the user-item matrix as additional users. This is similar to the approach taken by Tso-Sutter et al. (2008), who added tags as extra users and items in their approach. The fifth hybrid filtering approach is that of Basilico and Hofmann (2004). They performed movie recommendation and combined usage information with metadata collected from IMDB. Their hybrid approach used Joint Perceptron Ranking with different kernels representing the user correlations and metadata features to predict movie ratings. Basilico and Hofmann found that using metadata performed significantly worse than using usage information, but reported that this was due to the relatively low quality of the metadata collected.

There are at least three approaches that come closer to our own hybrid in the sense that they calculate user or item similarities based on extra information and plug those into traditional CF algorithms. First, Burke (2002) designed a hybrid restaurant recommender system that combines CF with a knowledge-based recommender system. They took the semantic ratings assigned to restaurants (e.g. 'expensive', 'lively', 'haute cuisine') and used those to calculate a semantic similarity between users. These user similarities were then plugged into a CF algorithm to generate recommendations. Second, Paulson and Tzanavari (2003) also come close to our own approach. They used the distance-based similarities between concepts in a domain ontology to determine the similarity between users, and plugged those similarities into a CF algorithm. Third, Ziegler et al. (2004) take a similar approach using a book taxonomy to extend the similarity calculation between users. All three of these approaches report significant improvements over traditional CF similarity metrics.

We present two findings with regard to the review of related work on hybrid filtering. First, while some researchers report significant improvements of their hybrid approaches over the separate algorithms, other researchers do not find any improvement at all. This makes it difficult to draw any definite conclusions as to the performance benefits of hybrid recommender systems. This inconclusiveness is caused partly by the second prevalent finding in the literature: the performance of content-based filtering depends strongly on (1) the quality of the metadata used and (2) the constructed features based on this metadata. Poor performance is most often attributed to low quality metadata.

5.5 Discussion

In this section, we step back and take stock of our findings with regard to exploiting metadata for social bookmarking recommendation. We start by discussing our findings with regard to our content-based filtering and hybrid filtering methods. We then describe the influence the different (combinations of) metadata fields have on recommendation performance. We end this section by discussing the results of our comparison with algorithms from the related work.

Content-based Filtering We first compared the two content-based filtering algorithms, profile-centric matching and post-centric matching. We found that a profile-centric approach, where we match the aggregated metadata of a user directly against the aggregated metadata of items, outperforms a similar approach at the post level. A likely explanation for the latter finding is that posts, as a lower level of information objects, carry much less metadata. Indeed, Table 5.1 shows that metadata coverage goes up as we proceed from the post level to the item and user levels of granularity. Metadata sparseness can be a problem for the post-centric approach: when most of the metadata fields are not filled for each post this means that some posts simply cannot be matched to other posts because there is not enough data. At the profile level, posts that lack certain metadata are combined with other posts that do have this metadata, ensuring more richly filled user and item representations, and subsequently better matching between profiles.

On the Delicious data set the post-centric approach performed better than the profile-centric approach. A possible reason for this is that the user profiles on Delicious show the greatest topical variety. Aggregating all posts into a single profile here might result in too broad a user representation, where there is always a part of the representation that matches some item. A more fine-grained approach might be able to overcome this. However, we do not see as strong an effect on BibBoo and do not have a clear explanation for this.

Hybrid Filtering When comparing our two hybrid filtering algorithms, user-based and item-based hybrid filtering, we found that item-based hybrid filtering performed best. However, it should be noted that user-based filtering was better in many of the individual field runs. We could expect the same explanation for the difference between profile-centric and post-centric to hold here: aggregation at a higher level suffers less from metadata sparseness. However, item profiles are a higher-level aggregation than posts, and coverage is indeed a bit higher for item representations. The DESCRIPTION fields in the BibBoo and BibArt data sets, for instance, have their coverage nearly doubled when aggregating post metadata into item representations.

Most importantly, we believe the algorithmic difference between user-based and item-based CF comes into play here. For user-based matching we calculate the user similarities based on content, but this content plays only a small role in generating the final recommendations. For the sake of argument, let us say that we only use the nearest neighbor for predictions (i.e., $N = 1$). We can expect certain items of the neighbor to be better recommendations than others, and indeed those items are what the active user and the neighbor user matched on so strongly. However, in selecting the items to be recommended no attention is paid to the actual topical overlap between the active user's items and the neighbor's items. Instead, each of the items of the active user's nearest neighbor is promoted equally, scored with the similarity between the users. In the item-based approach, there is a direct focus on the items of the active users and what other items in the data set they best match up with. If we hold N at 1 again, then new items are promoted directly according to their topical similarity with the active user's items. If the item similarity calculation is not hampered by sparsity as it was in usage-based CF, then we may expect this approach to generate better recommendations than the user-based approach. This argument also holds for larger neighborhood sizes.

Metadata Fields We observed that certain metadata fields provide better recommendations than other fields. The TITLE field, for instance, was found to perform worse on bookmark data sets than on article data sets. A likely explanation for this is that article titles are picked by the authors and are fixed to the articles. Aggregating the TITLE field into item profiles, for instance, will usually amount to mere duplications of the title. Users posting bookmarks, however, are free to change the title of the bookmark and are more likely to do so. These personalized versions of titles are less likely to match the right related items.

An interesting question is how much of an influence the sparseness of a metadata field has on performance. If we look at the correlation between coverage at the different levels of granularity and MAP, we find only weak correlations at the user level. Most metadata fields have high coverage at the user level, so it is mainly the quality of the field that has the biggest influence on the recommendation. At the item and user levels however, we find weak to moderate positive correlations ranging from 0.34 to 0.50. The latter finding suggests that

performance tends to go up as the coverage increases. Upon closer examination, we see that the fields with high sparsity always correspond to low performance. The fields with high coverage do not automatically achieve high MAP scores, suggesting that the usefulness for recommending related items and the quality of the metadata field play a larger role here, confirming the findings in the related work.

The influence of metadata quality on recommendation is also evident from the fact that in some cases the runs based on individual fields outperformed runs that combined all metadata fields. This shows that combining highly predictive fields with lower-quality fields can hurt performance and introduce too many false positives in the list of recommendations. We believe that feature selection techniques could be used here to determine the optimal set of metadata fields (Jain and Zongker, 1997). We found no significant differences between the runs with all intrinsic fields versus the runs with both intrinsic and extrinsic fields. This absence of significant differences implies that these supporting fields do not add anything to the recommendation process.

We cannot declare one of our four metadata-based algorithm as the best one. The optimal metadata-based algorithm depends on the data set and the metadata fields used. Based on our experimental results, however, we may conclude (1) that profile-centric matching and item-based hybrid filtering are most consistently among the best-performing algorithms, and (2) that using (at least) all of the intrinsic metadata fields combined typically yields the best performance.

Comparison to Folksonomic Recommendation In Section 5.4 we discussed a substantial body of related work for both our content-based and hybrid filtering approaches. We explained that our choice for an IR approach was inspired by the work on IMAs in the 1990s. A main difference is that, where they condensed item representations into the most salient keywords, we use the entire representation in the matching process. It is complicated, however, to compare findings from the early work on IMAs to those from our own work, because of the different conceptual designs and usage scenarios, and the temporal difference.

When comparing our metadata-based algorithms to the CF and TOBCF algorithms from Chapter 4, we found that our best metadata-based runs outperformed the best CF and TOBCF runs. Metadata-based runs performed best on three of our four data sets: BibBoo, BibArt, and CiteULike. We believe this is because using metadata results in richer representations for matching users and items. All of our best metadata runs use the combined metadata fields. On their own, each field can be seen as imperfect representations of the items and users, but combined they alleviate each others weak points and better represent the content than they do separately. This is in accordance with, for instance, the principle of polyrepresentation of Ingwersen (1994). However, we do not have an explanation as to why the CF runs performed better on the Delicious data set.

In addition, we compared the best GBCF algorithm from Section 4.6—the tag-aware fusion method of Tso-Sutter et al. (2008)—to our metadata-based algorithms in Subsection 5.3.3. The tag-aware fusion method outperforms our best meta-data algorithms on three of our data sets. Tag-aware fusion combines different representations—transaction patterns and tagging patterns—with different algorithms by fusing user-based and item-based filtering

predictions. In contrast, our metadata-based algorithms only combine different representations.

Because of the clear benefit of fusing different recommendation algorithms and different representation, we expect that combining different algorithms that use different parts of the data set to generate recommendations will lead to improvements over the component algorithms. In this light it would be interesting to combine the best CF runs with the best metadata-based runs. We will examine this and compare different combination methods in the next chapter.

An interesting question to ask is why one should consider other recommendation approaches, given that tag-aware fusion outperforms most of the other algorithms proposed so far. We can think of two reasons. The first reason is implementation effort. The metadata-based approaches are based on textual similarity matching as is commonly performed by search engines. The existing search infrastructure of a social bookmarking website could therefore be leveraged to implement metadata-based recommendation. However, the tag-aware fusion algorithm would have to be implemented from scratch. As the best metadata-based approaches are quite competitive compared to tag-aware fusion, we conjecture that it might be more efficient to implement metadata-based recommendation when mature search infrastructure is available. A second reason to choose another algorithm over tag-aware fusion could be running time of the algorithm. The similarity calculation phase in CF algorithms takes up the majority of the computational resources. As tag-aware fusion extends the normal ratings matrix $\mathbf{R}$ with two extra matrices $\mathbf{UT}$ and $\mathbf{IT}^T$, similarity computation also takes up more time because of the larger size of the combined matrices. This could be another reason to prefer one of the TOBCF algorithms, where the user and item similarities are calculated on only one of the matrices.

5.6 Chapter Conclusions and Answer to RQ 2

The metadata that users assign to their items is a rich source of information that could potentially improve recommendation performance. In this chapter we answered our second research question and its two subquestions.

RQ 2 How can we use the item metadata available in social bookmarking systems to provide accurate recommendations to users?

RQ 2a What type of metadata works best for item recommendation?

RQ 2b How does content-based filtering using metadata compare with folksonomic recommendations?

To address RQ 2, we proposed two types of recommendation methods that employ metadata: content-based filtering and hybrid filtering. Inspired by the work on IMAs, we took an IR approach to content-based filtering. We constructed textual representations of the

metadata assigned by users, assigned to items, and assigned to separate posts. We proposed two variants of content-based filtering, profile-centric matching and post-centric matching. The difference between the two variants is the level of granularity at which metadata representations are matched. We found that a profile-centric approach, where all of the metadata assigned by a user is matched against metadata representations of the items worked better than a post-centric approach that matched representations at a finer level of granularity. We believe this is due to sparseness issues of the fine-grained post-centric approach.

Our hybrid filtering method combined a standard nearest-neighbor CF algorithm with content-based filtering. Analogous to the standard CF algorithm we defined two variants: user-based hybrid filtering and item-based hybrid filtering. For both algorithms we use the metadata representations to calculate the user and item similarities respectively. These similarities are then used to locate the nearest neighbors to generate the recommendations according to the standard user-based and item-based CF algorithm. Of these two algorithms we found that item-based filtering with the metadata-derived similarities works best, as it did with tag overlap similarities.

When comparing all four metadata-based algorithms, we cannot single out a single-best combination of algorithm and metadata fields. Based on our experimental results, however, we may conclude (1) that profile-centric matching and item-based hybrid filtering are most consistently among the best-performing algorithms, and (2) that using (at least) all of the intrinsic metadata fields combined typically yields the best performance.

In answer to RQ 2a, we divided our 27 different metadata fields that occurred in our four data sets into two broad categories. The first category was item-intrinsic metadata, which relate directly to the content of the item being annotated. The second category was item-extrinsic metadata, which cover the administrative information that could never serve as a stand-alone source for recommendations. To determine the influence of the different types of metadata, we experimented with using different (combinations of) metadata fields. For each algorithm, we ran between 5 and 13 different experiments with different metadata fields, depending on the data set. We found that while sparsity of an individual metadata field does have an influence on recommendation performance, the quality of the information contained in the field is just as important. Based on our experimental results, we may conclude that combining all intrinsic metadata fields together tends to give the best performance, whereas extrinsic information, i.e., information not directly related to the content, does not.

Finally, to answer RQ 2b we compared our metadata-based approaches to the folksonomic recommendation algorithms proposed in Chapter 4. We found that recommending using metadata works better on three of our four data sets, making it a viable choice for recommendation despite being underrepresented in the related work so far.

Chapter 6

Combining Recommendations

In Chapters 4 and 5 we learned that combinations of different algorithms and representations tend to outperform individual approaches. Guided by our third research question, we examine this phenomenon in more detail in this chapter.

> **RQ 3** Can we improve performance by combining the recommendations generated by different algorithms?

The problem of effective item recommendation is too complex for any individual solution to capture in its entirety, and we expect that by combining different aspects of this problem we can produce better recommendations. Earlier, we saw evidence for the potential success of combination in the approach by Tso-Sutter et al. (2008) who successfully fused together the predictions of different algorithms using different representations of the data. We already proposed two combination approaches ourselves in the previous two chapters: our similarity fusion algorithm in Subsection 4.4.3 and our hybrid filtering algorithm in Subsection 5.2.2. On some data sets these produced superior recommendations, but the results were not conclusive. This naturally leads us to the following subquestion.

> **RQ 3a** What is the best recipe for combining the different recommendation algorithms?

In this chapter we will examine the possibilities of data fusion[1] in more detail. Instead of augmenting or combining features for recommendation, we will examine the effectiveness of combining the output of different recommendation runs (RQ 3), and compare the results against the other fusion variants we have proposed so far (RQ 3a).

Chapter 6 is organized as follows. We start in Section 6.1 by discussing related work on data fusion in the fields of recommender systems, IR and machine learning, and highlight some

[1] The term 'data fusion' can be ambiguous to a certain extent. In this chapter, we take it to mean output fusion, i.e., fusing the recommendation lists from different runs, analogous to the use of the term in the field of IR.

of the reasons why fusion is often successful. Then, in Sections 6.2 and 6.3, we describe our approach to fusing recommendations and which individual runs we select for this. Section 6.4 then describes our results, addressing RQ 3 and RQ 3a. In addition, we compare them to the similarity fusion and hybrid filtering techniques proposed in the previous two chapters (RQ 3a). We end with a discussion of the results in Section 6.5.

6.1 Related Work

We start in Subsection 6.1.1 by discussing related work on fusing different recommendation algorithms to improve upon the performance of the individual algorithms. Then, in Subsection 6.1.2, we discuss successful approaches to data fusion in two other fields: machine learning and IR. We conclude in Subsection 6.1.3 by looking at the underlying explanations for the success of data fusion to aid in the analysis of our own experiments with combining recommendations.

6.1.1 Fusing Recommendations

In the past decade, the field of recommender systems has already seen several different approaches to combining different recommendation algorithms. Burke (2002) presented a taxonomy of seven different methods for creating hybrid recommendation algorithms, which we reproduce here in Table 6.1. We briefly describe them below.

Table 6.1: A taxonomy of recommender system combination methods, as given by Burke (2002).

Hybridization method	Description
Mixed	Recommendations from several different recommenders are presented at the same time.
Switching	The system switches between recommendation techniques depending on the current situation.
Feature combination	Features from different recommendation data sources are thrown together into a single recommendation algorithm.
Cascade	One recommender refines the recommendations given by another.
Feature augmentation	The output from one technique is used as an input feature to another technique.
Meta-level	The model learned by one recommender is used as input to another.
Weighted	The scores of several recommendation techniques are combined together to produce a single recommendation.

The *mixed* hybridization method, arguably one of the most straightforward methods, presents all outputs of the different individual algorithms at the same time. The practical applicability of this technique is dependent on the scenario in which recommendations have to be produced; if a single results list is called for, then the recommendations will have to be merged. A *switching* hybrid algorithm switches between the different component algorithms based on certain criteria. For instance, the cold-start problem we mentioned in Subsection 2.1.1

could be a reason to base initial recommendations on the output of a content-based filtering algorithm. As soon as sufficient ratings are collected from the users, the system could then switch to a CF algorithm. In a *feature combination* approach, features from different types of algorithms (i.e., collaborative information, content-based, or knowledge-based) are combined and used as the input feature set for a single recommendation algorithm. Our similarity fusion approach from Subsection 4.4.3 is an example of a feature combination approach, as it uses both collaborative and topic-based information in the form of tags. Burke also describes two hybridization approaches that sequence two different recommendation algorithms. In the *cascaded* hybrid approach, one recommendation algorithm is first used to produce a coarse ranking of the candidate items, and the second algorithm then refines or re-ranks this candidate set into the final list of recommendations. In contrast, in a *feature augmented* hybrid algorithm one technique is employed to produce a rating of an item, after which that rating is used as an input feature for the next recommendation technique. As mentioned before, we view our hybrid filtering approach from Subsection 5.2.2 as a feature augmentation approach. The *meta-level* hybrid approach takes this a step further by using the entire model generated by the first algorithm as input for the second algorithm. Finally, a popular and straightforward way of combining algorithms is by producing a *weighted combination* of the output lists of the individual algorithms, where the different algorithms are all assigned separate weights.

While each of these seven combination methods has its merits and drawbacks, there has not been a systematic comparison of them on the same data sets using the same experimental setup. The lack of such a comparison makes it difficult to draw conclusions about which method is most effective in which situation. Most of the related work on recommender systems fusion has focused on combining content-based filtering with collaborative filtering. We discussed the most important of these approaches in Section 5.4, and these range from feature combination (Basu et al., 1998) and feature augmentation approaches (Mooney and Roy, 2000) to weighted combination algorithms. The latter combination method is one we will examine in more detail in this chapter, and the focus of the remainder of this related work subsection. For instance, Claypool et al. (1999) presented a weighted hybrid recommender system that calculated a weighted average of the output of two separate CF and content-based filtering components. The CF component receives a stronger weight as the data sets grows denser, gradually phasing out the influence of the content-based component. They did not find any significant differences between the performance of the separate components or the combined version. Pazzani (1999) combined three different recommendation algorithms: a CF algorithm, content-based filtering, and recommendation based on demographic information. They then used a majority-voting scheme to generate the final recommendations which increases precision over the individual approaches. Finally, the tag-aware fusion approach by Tso-Sutter et al. (2008) that we examined earlier, is also an example of a weighted hybrid recommender system, as it calculates a linearly weighted combination of separate user-based and item-based filtering predictions. This fusion approach was originally inspired by the work of Wang et al. (2006a), who formulated a generative probabilistic framework for the memory-based CF approach. Wang et al. generated the final ratings by not only fusing predictions based on three sources: (1) ratings of the same item by other users (i.e., user-based filtering); (2) different item ratings made by the same user (i.e., item-based filtering); and (3) data from other but similar users that rated other but

similar items. Their original model showed a significant performance increase compared to standard user-based and item-based filtering, and an improved resistance to ratings sparseness.

6.1.2 Data Fusion in Machine Learning and IR

Data fusion has been shown to improve performance not only in the field of recommender systems, but also in related fields. We discuss related work in two such fields: machine learning and IR.

Machine Learning In machine learning we see similar methods for combination and hybridization as those discussed by Burke (2002), but under different names. We discuss three of the most popular combination methods in machine learning. The first is *stacking*, where the output of one classifier serves as input feature for the next classifier (Wolpert, 1992). This method is known to be capable of recognizing and correcting recurring errors of the first-stage classifier, and corresponds to Burke's feature augmentation method. A second combination method is *bagging*, which generates multiple versions of a predictor by making bootstrap replicates of the original data sets (Breiman, 1996). The set of predictors are then used to obtain an aggregated predictor by doing majority voting when predicting an output class. Bagging is similar to Burke's weighted combination method. Third, *boosting* involves learning an optimally weighted combination of a set of weak classifiers to produce a strong classifier (Freund and Schapire, 1996). The process is iterative, as repeatedly misclassified instances receive a higher weight in the next round of learning the weak classifier. This way the entire data set can be covered correctly by the resulting strong classifier.

Information Retrieval Throughout this thesis we have taken an IR perspective on recommendation in our evaluation and algorithm design. We turn our attention to discussing related work on data fusion in IR in this section. An important distinction to make here is the one between *results fusion*, where the results of *different* retrieval algorithms on the *same* collection are combined, and *collection fusion*, where the results of one or more algorithms on *different* document collections are integrated into a single results list. We are not interested in the latter approach, and refer the interested reader to, for instance, Voorhees et al. (1995) for more information.

In IR, there are two prevalent approaches to results fusion: (1) combining retrieval runs that were generated using *different query representations* but with the same algorithm, or (2) combining retrieval runs that were generated using the same query, but with *different algorithms*. In our social bookmarking scenario, the first type of data fusion corresponds to using different representations of the user profile for recommendations—such as transaction patterns, tagging behavior, or assigned metadata—and then combining those different recommendation runs. The second approach corresponds to combining different recommendation algorithms—such as CF and content-based filtering—and fusing those predicted items into a single list of recommended items. Most approaches in IR also fall in one of these two categories, with some approaches spanning both. Our contribution in this chapter is that we will investigate the usefulness of both approaches for social bookmarking recommendation in answering RQ 3 and RQ 3a.

The earliest approaches to data fusion in IR stem from the 1990s when Belkin et al. (1993) investigated the effect of combining the result lists retrieved using different query representations of the same information need. They showed that progressive combination of query formulations leads to progressively improving retrieval performance, and extended their own work in Belkin et al. (1995) with an additional set of experiments confirming their earlier findings. Later work on combining different query and document representations includes the work by Ingwersen and Järvelin (2005), who view the fusion problem from a cognitive IR perspective. They formulated the principle of *polyrepresentation* in which each query or document representation, searcher, and retrieval model can be seen as a different representation of the same retrieval process (Ingwersen, 1994, 1996). The validity of this principle has been confirmed for queries, documents, searchers, and retrieval algorithms in Skov et al. (2008) and Larsen et al. (2009).

Some of the earliest work on fusing the results of different retrieval algorithms includes Croft and Thompson (1987), who fused a probabilistic retrieval model with a vector space model. Bartell et al. (1994) also examined results fusion using different retrieval algorithms. They proposed a linear combination of retrieval runs using different variants of the same IR algorithm, and showed significant improvements over the individual runs. Vogt and Cottrell (1998) later revisited this work and used linear regression to determine the optimal combination of run weights. Fox and Shaw (1994) investigated a set of unweighted combination methods that have become standard methods for data fusion in IR. They tested three basic combination methods CombMAX, CombMIN, and CombMED that respectively take the maximum, minimum, and median similarity values of a document from among the different runs. In addition, they also proposed three methods CombSUM, CombMNZ, and CombANZ that have consistently shown to provide good data fusion results. The CombSUM method fuses runs by taking the sum of similarity values for each document separately; the CombMNZ and CombANZ methods do the same but respectively boost and discount this sum by the number of runs that actually retrieved the document. Fox and Shaw (1994) showed that the latter three methods were among the best performing fusion methods. This work was later extended by Lee (1997) with a more thorough analysis, and they found that CombSUM and CombMNZ were again the best-performing methods. Periodically, these combination methods have been re-examined in different settings, such as monolingual retrieval (Kamps and De Rijke, 2004), or against different probabilistic fusion methods (Croft, 2000; Aslam and Montague, 2001; Renda and Straccia, 2003). The CombSUM and CombMNZ methods have been shown to consistently improve upon the performance of the individual retrieval runs.

6.1.3 Why Does Fusion Work?

In this section, we introduce various reasons that have been proposed in the related work to explain the success of data fusion. Because of our IR perspective on recommendation, we focus exclusively on explanations from this field. We will use these explanations later on in the analysis of our experiments to explain what is happening in recommender systems fusion.

Belkin et al. (1993) argue that the success of query result fusion is due to the fact that the problem of effective representation and retrieval is so complex that individual solutions can never capture its complexity entirely. By combining different representations and retrieval models, more aspects of this complex situation are addressed and thus more relevant documents will be retrieved. This is similar to the explanation from the polyrepresentation point of view (Ingwersen and Järvelin, 2005), which states that using different representations and retrieval models will retrieve different sets of information objects from the same collection of objects with a certain amount of overlap. Merging cognitively different representations and retrieval models corresponds to modeling different aspects of the task as suggested by Belkin et al. (1993), and the overlapping documents are therefore seen as more likely to be relevant.

The latter effect of overlapping documents having a higher likelihood of being relevant was dubbed the *Chorus* effect by Vogt and Cottrell (1998). The *Chorus* effect is also related to what Vogt and Cottrell define as the *Skimming* effect:

> *The Skimming Effect happens when retrieval approaches that represent their collection items differently may retrieve different relevant items, so that a combination method that takes the top-ranked items from each of the retrieval approaches will push non-relevant items down in the ranking.*

A third, contradictory explanation by Vogt and Cottrell for the success of fusion is the *Dark Horse* effect, which states that certain retrieval models might be especially suited to retrieving specific types of relevant items compared to other approaches.

Spoerri (2007) describes the two of these effects, the *Chorus* and *Skimming* effects, under different names: the *Authority* effect and the *Ranking* effect. The *Authority* effect describes the phenomenon that the potential relevance of a document increases as the number of systems retrieving it increases, while the *Ranking* effect describes the observation that documents higher up in ranked lists and found by more systems are more likely to be relevant. They provide empirical evidence for these effects as the cause for the success of fusion. In addition, they show that the two effects can be observed regardless of the type of query representation.

It is clear from all of the above definitions that it is possible for one effect to conflict with another. For instance, although one algorithm might be particularly well-suited for retrieving specific types of relevant items (i.e., the *Dark Horse* effect), they might be pushed down too far in the ranking by other items, relevant or not, that occur in multiple retrieval runs (i.e., the *Chorus* effect).

6.2 Fusing Recommendations

When combining the output of different recommendation algorithms, a decision needs to be made about what to combine: the scores or ratings assigned to the recommended items, or the ranks of the items in the list of recommendations. These two options are commonly

referred to as *score-based fusion* and *rank-based fusion* in the related work. Earlier studies reported on the superiority of using retrieval scores over document ranks for data fusion (Lee, 1997), but later studies have re-examined this and found few significant differences between the two (Renda and Straccia, 2003). We opt for using score-based fusion in our experiments, since we could find no conclusive evidence to suggest that rank-based fusion is better. The decision between score-based and rank-based fusion can also be seen as a decision of what should be normalized: the item ranks or the item scores. In the field of IR, different retrieval runs can generate wildly different ranges of similarity values, so a normalization method is typically applied to each retrieval result to map the score into the range $[0,1]$. We find the same variety in score ranges when fusing different recommendation runs, so we also perform normalization of our recommendation scores. Typically, the original recommendation scores $score_{original}$ are normalized using the maximum and minimum recommendation scores $score_{max}$ and $score_{min}$ according to the formula proposed by Lee (1997):

$$score_{norm} = \frac{score_{original} - score_{min}}{score_{max} - score_{min}}. \tag{6.1}$$

Several other normalization methods have also been proposed, such as Z-score normalization and Borda rank normalization (Aslam and Montague, 2001; Renda and Straccia, 2003). However, none of these methods have been proven to result in significantly better performance, so we use simple score-based normalization according to Equation 6.1.

We introduced six standard fusion methods in our discussion of the related work in Subsection 6.1.2. We select the three methods for our experiments that have shown the best performance in related work: CombSUM, CombMNZ, and CombANZ. These standard combination methods are defined as follows. Let us consider a set of N different recommendation runs R for a specific user that we want to fuse together. Each run r_n in the set R consists of a ranking of items, and each item i has a normalized recommendation score $score(i, r_n)$ in that run r_n. Let us also define the number of hits of an item in R as $h(i,R) = |\{r \in R : i \in r\}|$, i.e., the number of runs that i occurs in. We can then represent all three combination methods CombSUM, CombMNZ, and CombANZ using the following equation:

$$score_{fused}(i) = h(i,R)^{\gamma} \cdot \sum_{n=1}^{N} score(i, r_n). \tag{6.2}$$

The γ parameter governs which combination method we use and can take one of three values. Setting γ to 0 is equal to the CombSUM method, where we take the sum of the scores of the individual runs for an item i. For CombMNZ, we take the sum of the scores of the individual runs for an item i, multiplied by the number of hits of an item $h(i,R)$. Here, γ is set to 1. Finally, setting γ to -1 results in the CombANZ combination method, where we take the sum of the scores of the individual runs for an item i and divide it by the number of hits of an item $h(i,R)$. In other words, we calculate the average recommendation score for each item.

These three combination methods are all unweighted, i.e., each run has an equal weight in the fusion process. However, a different common fusion approach in both recommender systems and IR research is to do a *linear combination* of the individual runs as proposed by Bartell et al. (1994) and Vogt and Cottrell (1998). The benefit of weighting different runs separately is obvious: not every run exhibits the same level of performance, and would therefore not be assigned the same weight in the optimal combination. When we linearly combine runs, each run is assigned a preference weight w_n in the range $[0,1]$. We also test weighted versions of the CombSUM, CombMNZ, and CombANZ combination methods. The weighted versions of the methods are defined as:

$$score_{fused}(i) = h(i,R)^{\gamma} \cdot \sum_{n=1}^{N} w_n \cdot score(i,r_n). \tag{6.3}$$

In the situations where we combine the results of two or more recommendation runs, the optimal combination weights could be determined by a simple exhaustive parameter sweep, as we did for our similarity fusion approach in Subsection 4.4.3. When combining more than two runs, however, performing an exhaustive search for the optimal weights quickly becomes intractable, as it is exponential in the number of weights. We therefore used a random-restart hill climbing algorithm to approximate the optimal weights for all our fusions runs (Russel and Norvig, 2003). We randomly initialized the weights for each run, then varied each weight between 0 and 1 with increments of 0.1. We selected the value for which the MAP score is maximized and then continued with the next weight. The order in which run weights were optimized was randomized, and we repeated the optimization process until the settings converged. We repeated this process 100 times, because the simple hill climbing algorithm is susceptible to local maxima. We used our 10-fold cross-validation setup to determine these optimal weights. We then selected the weights that result in the best performance and generated the recommendations on our final test set using these optimal weights.

6.3 Selecting Runs for Fusion

After deciding *how* to fuse recommendation runs together in the previous section, we then need to determine *which* runs we should fuse together. We let ourselves be guided here by the intuition brought forward in the work of Belkin et al. (1993) and Ingwersen and Järvelin (2005), who argue that recommendations generated using cognitively dissimilar representations and algorithms, i.e., that touch upon different aspects of the item recommendation process yield the best fused results. We consider two aspects of recommendation in our selection of recommendation runs: representations and algorithms. For instance, we consider item-based filtering runs that use transaction patterns as a source of item similarity to be the same algorithmically as item-based filtering runs that use tag overlap similarities, but different in the way they represent the users and items in the system. Our two hybrid filtering approaches on the other hand use the same metadata representation of the system

content, but are different algorithms[2]. We do not consider fusing runs that do not differ on at least one of these dimensions. For instance, combining two user-based filtering runs that both use transaction patterns for finding the nearest neighbors and only differ in the type of weighting is not likely to result in improved recommendations after fusion. Table 6.2 shows which fusion experiments we perform.

Table 6.2: An overview of our ten fusion experiments. The second and third columns denote if the fusion experiment fuses together runs using different representations or different algorithms respectively.

Run ID	Diff. repr.	Diff. alg.	Description
Fusion A	No	Yes	Best user-based and item-based filtering runs based on usage similarity (from Section 4.3).
Fusion B	No	Yes	Best user-based and item-based filtering runs based on tag overlap similarity (from Subsection 4.4.1).
Fusion C	Yes	Yes	Best usage-based and tag overlap runs together.
Fusion D	No	Yes	Best content-based filtering runs (from Subsection 5.2.1).
Fusion E	No	Yes	Best user-based and item-based filtering runs based on metadata-based similarity (from Subsection 5.2.2).
Fusion F	Yes	Yes	Best content-based and hybrid filtering runs together.
Fusion G	Yes	Yes	Best folksonomic and metadata-based runs together.
Fusion H	Yes	Yes	All four best runs from fusion experiments A and B together.
Fusion I	Yes	Yes	All four best runs from fusion experiments D and E together.
Fusion J	Yes	Yes	All eight best runs from fusion experiments A, B, D, and E.

In the first seven of our fusion experiments we fuse only pairs of best-performing runs within a specific set of runs. From Chapter 4, for instance, we combine the best user-based and item-based runs for the similarity representations: usage-based and tag overlap. We also combine the best runs of each representation type. We perform three similar fusion experiments for Chapter 5. Fusion experiment G then combines the best runs from Chapter 4 and 5 together. In addition to these seven pairwise fusion runs, we also experiment with fusing the best four runs from each chapter together in fusion experiments H and I respectively. Finally, we fuse all eight best runs from chapters 4 and 5 together. This results in a total of ten experiments.

6.4 Results

Table 6.3 lists the results of our fusion experiments. What we see is that, overall, fusing recommendation results is successful: in 36 out of 40 fusion runs we find a performance increase over the best individual runs. The best fusion runs on our four data sets show improvements ranging from 14% to 73% in MAP scores. When we take a look at the three different fusion methods, we see that CombSUM and CombMNZ both consistently provide good performance, and that they outperform each other in half of the cases. The difference between the two is never significant. In contrast, the CombANZ method performs poorly across the board: only on the Delicious data set does it achieve reasonable performance in

[2]Even though they are both memory-based algorithms, we consider user-based and item-based filtering to be different algorithms.

Table 6.3: Results of our fusion experiments. Reported are the MAP scores and the best-performing fusion methods for each set of fusion experiments are printed in bold. Boxed runs are the best overall. Significant differences are calculated over the best individual runs of each fusion run. The percentage difference between the best fusion experiment and the best individual run from the previous chapters is indicated in the bottom row.

Run	Method	bookmarks		articles	
		BibSonomy	Delicious	BibSonomy	CiteULike
Fusion A	CombSUM	0.0282	0.0050	0.0910	0.0871
	CombMNZ	0.0249	**0.0065**	0.0924	0.0871
	CombANZ	0.0175	0.0043	0.0687	0.0691
	Weighted CombSUM	**0.0362**	0.0056	0.0995	**0.0949**$^{\triangle}$
	Weighted CombMNZ	0.0336	**0.0065**	**0.1017**	0.0947$^{\triangle}$
	Weighted CombANZ	0.0303	0.0043	0.0924	0.0934$^{\triangle}$
Fusion B	CombSUM	0.0360	0.0024	0.1062	0.0788
	CombMNZ	0.0350	0.0032	0.1104	0.0801
	CombANZ	0.0245	0.0023	0.0904	0.0560$^{\triangledown}$
	Weighted CombSUM	0.0374	0.0102	0.1171	0.0945$^{\triangle}$
	Weighted CombMNZ	**0.0434**	0.0093	**0.1196**	**0.0952**$^{\triangle}$
	Weighted CombANZ	0.0314	**0.0105**	0.1028	0.0798
Fusion C	CombSUM	0.0424	0.0102	0.1543	0.1235$^{\blacktriangle}$
	CombMNZ	0.0389	0.0061	0.1453	0.1239$^{\blacktriangle}$
	CombANZ	0.0229	0.0057	0.0787	0.0896
	Weighted CombSUM	**0.0482**	0.0109	**0.1593**	0.1275$^{\blacktriangle}$
	Weighted CombMNZ	0.0477	**0.0115**	0.1529	**0.1278**$^{\blacktriangle}$
	Weighted CombANZ	0.0305	0.0089	0.1262	0.0973
Fusion D	CombSUM	0.0322	0.0020	0.1273	0.0883$^{\triangledown}$
	CombMNZ	0.0320	0.0021	0.1273	0.0884$^{\triangledown}$
	CombANZ	0.0257	0.0013	0.0142	0.0112$^{\blacktriangledown}$
	Weighted CombSUM	**0.0388**	0.0035	**0.1303**	0.1005
	Weighted CombMNZ	0.0387	0.0037	0.1302	**0.1008**
	Weighted CombANZ	0.0311	**0.0038**	0.1127	0.0371
Fusion E	CombSUM	0.0410	**0.0051**	0.1314	0.0889
	CombMNZ	0.0371	0.0037	0.1349	0.0926$^{\triangle}$
	CombANZ	0.0247	0.0036	0.0636	0.0464$^{\triangledown}$
	Weighted CombSUM	**0.0514**	**0.0051**	0.1579	0.0908$^{\blacktriangle}$
	Weighted CombMNZ	0.0473	0.0043	**0.1596**	**0.0945**$^{\blacktriangle}$
	Weighted CombANZ	0.0323	0.0042	0.1028	0.0636$^{\blacktriangledown}$
Fusion F	CombSUM	0.0418	0.0049	0.1590	0.1117$^{\blacktriangle}$
	CombMNZ	0.0415	0.0050	0.1593	0.1099$^{\triangle}$
	CombANZ	0.0142	0.0023	0.0313$^{\triangledown}$	0.0284$^{\blacktriangledown}$
	Weighted CombSUM	0.0492	0.0051	**0.1600**	0.1127$^{\blacktriangle}$
	Weighted CombMNZ	**0.0494**	**0.0056**	0.1599	**0.1136**$^{\triangle}$
	Weighted CombANZ	0.0379	0.0038	0.1475	0.0699
Fusion G	CombSUM	0.0470	0.0051	0.1468	0.1511$^{\blacktriangle}$
	CombMNZ	0.0472	0.0046	0.1404	0.1448$^{\blacktriangle}$
	CombANZ	0.0235	0.0051	0.1368	0.0084$^{\blacktriangledown}$
	Weighted CombSUM	0.0524	0.0102	**0.1539**	**0.1556**$^{\blacktriangle}$
	Weighted CombMNZ	**0.0539**	**0.0109**	0.1506	0.1478$^{\blacktriangle}$
	Weighted CombANZ	0.0421	0.0098	0.1430	0.0866
Fusion H	CombSUM	0.0441	0.0049	0.1137	0.1064
	CombMNZ	0.0463	0.0047	0.1129	0.1117$^{\blacktriangle}$
	CombANZ	0.0134	0.0041	0.0627	0.0540$^{\triangledown}$
	Weighted CombSUM	**0.0619**	0.0077	**0.1671**	0.1276$^{\blacktriangle}$
	Weighted CombMNZ	0.0616	**0.0092**	0.1409	**0.1286**$^{\blacktriangle}$
	Weighted CombANZ	0.0247	0.0069	0.1063	0.0901
Fusion I	CombSUM	0.0507	0.0042	0.1468	0.1057
	CombMNZ	0.0502	0.0035	0.1479	0.1077
	CombANZ	0.0171	0.0027	0.0049	0.0084$^{\blacktriangledown}$
	Weighted CombSUM	**0.0565**	**0.0065**	**0.1749**	**0.1188**$^{\blacktriangle}$
	Weighted CombMNZ	0.0559	0.0052	0.1716	0.1157$^{\triangle}$
	Weighted CombANZ	0.0307	0.0033	0.1206	0.0454$^{\blacktriangledown}$
Fusion J	CombSUM	0.0507	0.0056	0.1617	0.1211
	CombMNZ	0.0502	0.0062	0.1613	0.1260$^{\triangle}$
	CombANZ	0.0085	0.0027	0.0163$^{\triangledown}$	0.0163$^{\triangledown}$
	Weighted CombSUM	0.0681	**0.0090**	**0.1983**$^{\triangle}$	**0.1531**$^{\blacktriangle}$
	Weighted CombMNZ	**0.0695**	0.0086	0.1904$^{\triangle}$	0.1381$^{\blacktriangle}$
	Weighted CombANZ	0.0309	0.0039	0.0954	0.0589
% Change over best individual run		+72.9%	+13.9%	+31.3%	+57.6%

some cases. The CombANZ method performs especially poorly on the CiteULike data set: in many cases it performs significantly worse than the individual runs. When we compare the unweighted combination methods against the weighted combination method, we find that the latter consistently outperform the unweighted fusion approaches. In some cases, these differences are statistically significant, as is the case in fusion run J for BibArt and CiteULike. Here, the weighted CombSUM runs achieve the data set-best MAP scores of 0.1983 and 0.1556 respectively, which are significantly higher than the unweighted CombSUM runs at 0.1617 ($p = 0.04$) and 0.1211 ($p = 1.9 \cdot 10^{-6}$). This confirms the findings of others such as Vogt and Cottrell (1998) and Kamps and De Rijke (2004) who found similar advantages of weighted combination methods. Typically, the best performing component runs are assigned the higher weights than the other runs. The optimal weights for the 120 different weighted combination runs are included in Appendix C.

We considered two different recommendation aspects when deciding which runs to combine: representations and algorithms. Pairwise fusion runs A, B, D, and E all combine different algorithms that use the same representations, whereas runs C, F, and G combine pairs of runs that vary both in representation and algorithm. The results show that the latter type of fusion, where different recommendation aspects are combined consistently performs better than when only one of the aspects is varied among the paired runs. We observe the same thing for runs H, I, and J, that combine more than two runs at the same time. Run J combines eight different runs that represent six different algorithms and four different types of representations, and achieves the best overall MAP scores on three of our four data sets. On the Delicious data set though, we find that increasing the number of runs to be combined is not always beneficial: we see better performance on the pairwise B, C, and G fusion runs than on the fusion runs that combine more individual runs.

In the next subsection we extend our analysis from determining which weighted combination methods work best to why these methods provide such superior performance (Subsection 6.4.1). Then, in Subsection 6.4.2, we compare the best weighted combination methods to the similarity fusion and hybrid filtering techniques introduced in Subsection 4.4.3 and Subsection 5.2.2 respectively.

6.4.1 Fusion Analysis

While it is useful to determine exactly which combination methods provide the best performance, it would be equally interesting and useful to find out what it is exactly that makes fusion outperform the individual runs. Belkin et al. (1993) provides two different rationales for the success of fusion approaches that we already mentioned in Section 6.1. The first is a *precision-enhancing effect*: when multiple runs are combined that model different aspects of the task, the overlapping set of retrieved items are more likely to be relevant. In other words, more evidence for the relevance of an item to a user translates to ranking that item with higher precision. The second rationale is a *recall-enhancing effect*, and describes the phenomenon that multiple runs that model different aspects will retrieve different set of relevant items. Fusing these individual runs can then merge these sets and increase recall of the relevant items.

Let us zoom in on two of the more successful combination runs, and analyze why we see the improvements that we do. We select two runs that significantly improve over the individual runs: (1) fusion run G for CiteULike, where we combine the best folksonomic recommendation run with the best metadata-based run, and (2) fusion run J for BibArt, where we combine all eight best individual runs from A, B, D, and E. Both runs combine different algorithms and different representation types. In our analysis we take an approach similar to Kamps and De Rijke (2004) who analyzed the effectiveness of different combination strategies for different European languages. We manipulate our results in two different ways before the MAP scores are calculated to highlight the improvements in precision and recall due to fusion. Each time we compare the fused run with each of the individual runs separately to determine the effects of those runs.

For identifying the enhancements due to increased precision, we ignore the ranking of items that did not occur in the individual run in our calculation of the MAP scores. This neutralizes the effect of additionally retrieved items and isolates the contribution of items receiving a better ranking (Kamps and De Rijke, 2004). Table 6.4 shows the results of our analysis for the two fusion runs, and the fourth column shows the MAP scores attributed to better ranking of the items that were originally present. For example, if we disregard the additionally retrieved items from the fused run in the top table, and only look at the relevant items already retrieved by run 2, we see that for run 2 we get a MAP score of 0.1546. This is an increase of 56.6% in MAP score over the original MAP score of 0.0987, which is due to a better ranking of the items retrieved by run 2.

Table 6.4: Results of our fusion analysis. For each individual run we report the original MAP score, the total number of relevant retrieved items by the run, the MAP score attributed to enhanced precision, and the MAP score attributed to enhanced recall.

CiteULike – Fusion run G – Weighted CombSUM

Run	Original MAP	Relevant docs	Precision-enh. MAP		Recall-enh. MAP	
Run 1	0.0887	579	0.1343	+51.3%	0.1084	+22.1%
Run 2	0.0987	791	0.1546	+56.6%	0.0992	+0.5%
Fused run	0.1556	791	-		-	

Bibsonomy bookmarks – Fusion run J – Weighted CombSUM

Run	Original MAP	Relevant docs	Precision-enh. MAP		Recall-enh. MAP	
Run 1	0.0853	86	0.1889	+121.5%	0.0937	+9.8%
Run 2	0.0726	92	0.1501	+106.7%	0.1071	+47.5%
Run 3	0.0452	55	0.1315	+190.9%	0.1123	+148.5%
Run 4	0.1084	55	0.1355	+25.0%	0.1685	+55.4%
Run 5	0.1261	115	0.1981	+57.1%	0.1263	+0.2%
Run 6	0.1173	111	0.1970	+67.7%	0.1186	+1.1%
Run 7	0.0404	66	0.1709	+323.0%	0.0638	+57.9%
Run 8	0.1488	90	0.1859	+24.9%	0.1591	+6.9%
Fused run	0.1983	115	-		-	

For identifying the enhancements due to increased recall, we look at the contributions the items newly retrieved by the fusion run make to the MAP score. This means we treat the items retrieved by an individual run as occurring at those positions in the fused run as well. Any increases in MAP are then due to newly retrieved items that were not present in the

individual run, i.e., increased recall, and not due to a better ranking because of combined evidence (Kamps and De Rijke, 2004). These MAP scores are listed in the sixth column in Table 6.4. For example, if we consider only the additionally retrieved items from the fused run in the top table compared to run 1, we see that for run 1 we get a MAP score of 0.1084. This is an increase of 22.1% in MAP score over the original MAP score of 0.0887, which is due to the improved recall of the fusion run.

The results in Table 6.4 show that combining different runs increases the number of relevant items retrieved by the combination run (third column). However, this increased recall does not necessarily mean that the improvement in the MAP scores is also due to these additionally retrieved items. We see from the adjusted MAP scores that both precision- and recall-enhancing effects are present. However, fusion clearly has a stronger effect on increasing the precision of the recommendations, and the increases in MAP score are almost always due to a better ranking of the documents. These results for these two fusion runs are representative for other fusion runs that (significantly) improved over their component runs. In addition, our findings confirm those of Kamps and De Rijke (2004): most of the effects of fusion they observed were also due to the improved ranking of the documents.

6.4.2 Comparing All Fusion Methods

Earlier in this thesis, in Subsections 4.4.3 and 5.2.2, we already proposed two other fusion approaches. The first, similarity fusion, was a feature combination approach and involved fusing two similarity matrices together in their entirety. The second, hybrid filtering, was a feature augmentation approach and used content-based user and item similarities in a CF algorithm. How do these two approaches stack up against our weighted fusion runs from this chapter? Table 6.5 compares the best fusion runs of this chapter for each data set against the best runs of the other two fusion approaches.

Table 6.5: Comparison of our three different approaches to recommender systems fusion. Reported are the MAP scores and the best-performing fusion method for for each data set is printed in bold. Significant differences are calculated between the best and second-best runs for each data set.

Run	bookmarks		articles	
	BibSonomy	Delicious	BibSonomy	CiteULike
Similarity fusion (from Section 4.4.3)	0.0350	0.0102	0.1210	0.0791
Hybrid filtering (from Section 5.2.2)	0.0399	0.0039	0.1510	0.0987
Weighted run fusion	**0.0695**	**0.0115**	**0.1983**$^{\triangle}$	**0.1531**$^{\blacktriangle}$
% Change over second best run	+74.2%	+12.7%	+31.3%	+55.1%

We can clearly see that taking a weighted combination of recommendation runs is superior to the other two approaches. This difference is significant on both the BibArt and CiteULike data sets, and the weighted run fusion approach is also significantly better than the hybrid filtering approach on the BibBoo data set.

Finally, we would like to remark that, except for the Delicious data set, weighted run fusion outperforms the tag-aware fusion approach of Tso-Sutter et al. (2008). These improvements are statistically significant ($p < 0.05$) on the BibArt and CiteULike data set.

6.5 Discussion & Conclusions

We found that combining different recommendation runs yields better performance compared to the individual runs, which is consistent with the theory behind data fusion and with the related work. Weighted fusion methods consistently outperform their unweighted counterparts. This is not surprising as it is unlikely that every run contributes equally to the final result, and this was also evident from the optimal weight distribution among the runs.

In addition, we observed that combination methods that reward documents that show up in more of the base runs—CombSUM and CombMNZ—are consistently among the best performers. In contrast, the CombANZ method performed worse than expected on our data sets. One reason for this is that CombANZ calculates an average recommendation score across runs for each item. There is no bonus for items that occur in multiple runs such as CombSUM and CombMNZassign, and run overlap is an important indicator of item relevance. In addition, the averaging of CombANZ can lead to exceptionally performing base runs being snowed under; this is especially apparent for the fusion experiments where four and eight runs were combined. When more runs are combined, CombANZ starts performing worse, relative to the performance of pairwise fusion runs with CombANZ.

A third finding from our fusion experiments was a confirmation of the principle put forward by Ingwersen and Järvelin (2005) and Belkin et al. (1993): it is best to combine recommendations generated using cognitively dissimilar representations and algorithms, touching upon different aspects of the item recommendation process.

We explored two different aspects to recommendation, representation and the choice of algorithm, and indeed found that runs that combine multiple, different recommendation aspects perform better than runs that consider variation in only one recommendation aspect. We also observed that for two data sets, BibBoo and BibArt, combining more runs tends to produce better performance. Here, the best performing fusion runs were those that combined eight base runs that varied strongly in the type of algorithm and representation.

A separate analysis confirmed that most of the gains achieved by fusion are due to the improved ranking of items. When multiple runs are combined, there is more evidence for the relevance of an item to a user, which translates to ranking that item with higher precision. Improved recall plays a much smaller role in improving performance. Overall, we find strong evidence for the *Chorus effect* in our experiments and, to a lesser extent, support for the *Ranking effect*. The lack of recall-related improvements suggests that we do not see the *Dark Horse effect* occurring in our fusion experiments.

6.6 Chapter Conclusions and Answer to RQ 3

We observed earlier in Chapters 4 and 5 that combining different algorithms and representations tends to outperform the individual approaches. Guided by our third research question and its subquestion, we examined this phenomenon in more detail in this chapter.

RQ 3 Can we improve performance by combining the recommendations generated by different algorithms?

RQ 3a What is the best recipe for combining the different recommendation algorithms?

We found a positive answer to RQ 3: combining different recommendation runs yields better performance compared to the individual runs on all data sets. In answer to RQ 3a, we identified several ingredients for successfully combining recommendation algorithms, such as combining approaches that cover different aspects of the item recommendation task. By combining different algorithms and different representations of the data, we achieved the best results. We compared weighted fusion methods with unweighted fusion methods and found that weighted methods performed best. This is understandable, since not every run contributes equally to the final result. Another ingredient for successful fusion is using a combination method that rewards documents that show up in more of the individual runs, harnessing the *Chorus* and *Authority* effects. After a detailed analysis, we learned that these performance improvements were largely due to a precision-enhancing effect: the ranking of overlapping items improves when they are retrieved by more runs. While fusion also increases recall, this has only a weak effect on performance.

II

Growing Pains: Real-world Issues in Social Bookmarking

So far in this thesis, we have focused on recommendation in the social bookmarking scenario from a rather idealized perspective. All of the experiments described in the previous chapters were performed on snapshots of the respective social bookmarking websites. We froze the situation at a certain point in time to enable a fair comparison between the different recommendation algorithms. In real life, however, a social bookmarking website is far from static. Delicious, for instance, is one of the most popular social bookmarking services and received an average of 140,000 posts per day in 2008 according to the independently sampled data collected by Philipp Keller[3].

In addition to freezing our datasets, we also implicitly assumed that all items in our datasets were unique and of equal quality, at least from the viewpoint of the recommendation algorithm. However, with the dynamic growth of social bookmarking also emerged its growing pains. One such problem is spam, and social bookmarking services—as is inevitable for nearly any website supporting user interaction—also suffer from this problem. Here, malicious users add items designed to mislead, or items that the site's genuine users do not wish to receive. Obviously, these spam items should never be recommended to any user, but how serious is this problem? What is the proportion of spam items on social bookmarking websites and does it pose a problem for recommendations if these spam items are not detected, demoted, or prevented? A second possible problem for social bookmarking is the presence of duplicates. Users rarely take the time to check if the items they posted already exist on the social bookmarking website and many websites do not offer any user interface mechanisms for detecting duplicates. However, the question may be raised: does the presence of duplicate content dilute the connections between users? And if so, (1) how can we detect these duplicates, and (2) how does this influence the recommendation process?

In Part II of this thesis, we will broaden our narrow perspective on recommendation and investigate the effects of the aforementioned growing pains of social bookmarking from a recommendation perspective. Both issues are addressed separately in the next two chapters and come with their own research questions.

[3] Available at http://deli.ckoma.net/stats; last visited January 2009.

RQ 4 How big a problem is spam for social bookmarking services?

RQ 5 How big a problem is the entry of duplicate content for social bookmarking services?

This part of the thesis is organized as follows. In Chapter 7 we look at how pervasive spam is in the social bookmarking services. Moreover, we focus on how we can detect such content automatically, and what kind of effect spam has on the recommendation process. Chapter 8 follows a similar path in assessing the problem of duplicate content.

Chapter 7

Spam

The term 'spam' was originally used to refer to the abuse of electronic messaging systems that started in the mid-1990s on Usenet newsgroups, and quickly crossed over to e-mail messaging. According to conservative estimates, in the first half of 2008 around 85% of all e-mail sent in the world was spam[1]. The notion of spam is subjective by nature, but we define it here as content designed to mislead the perceiver, or content that the legitimate users of a system, site, or service do not wish to receive. Motivation for spamming can range from making a profit through advertising and self-promotion to disruption and disparagement of competitors (Heymann et al., 2007). Spamming is economically viable because joining the abused systems and injecting spam content into them is straightforward in general. Spamming also requires virtually no operating costs beyond the management of the automatic spamming software. In addition, it is often difficult to hold spammers accountable for their behavior.

Any system that relies on user-generated content is vulnerable to spam in one form or another. Indeed, many other electronic systems that allow users to store, share, and find online resources have also come under attack from spamming attempts in recent years. Search engines, for instance, suffer increasingly from so-called *spamdexing* attempts with content especially created to trick search engines into giving certain pages a higher ranking than they deserve (Gyöngyi and Garcia-Molina, 2005). Spam comments are also becoming an increasingly bigger problem for websites that allow users to react to content, such as blogs and video and photo sharing websites (Mishne and de Rijke, 2006). Finally, Wikipedia—another system focusing around user-generated content—has also seen an increase in research interest in automatic approaches to spam detection (Priedhorsky et al., 2007).

Social websites and social bookmarking services have become an increasingly popular part of the Web, but their focus on user-generated content also makes them vulnerable to spam, threatening their openness, interactivity, and usefulness (Heymann et al., 2007). In this

[1] According to http://www.maawg.org/about/MAAWG_2008-Q2_Metrics_Report9.pdf, last visited July 21, 2009.

thesis, we recognize spam to be one of the growing pains social bookmarking systems experience as they attract more users. In this chapter, we wish to find out how big of a problem spam is for social bookmarking and whether we can automatically detect spam in social bookmarking systems. We will also investigate whether and what kind of effect the presence of spam has on recommendation. Are anti-spam measures necessary to protect recommendation quality? Or do spammers form a separate clique in the social network that has no influence on the recommendation process? This leads us to our fourth research question and two, more specific subquestions.

RQ 4 How big a problem is spam for social bookmarking services?

RQ 4a Can we automatically detect spam content?

RQ 4b What influence does spam have on the recommendation performance?

To answer these questions we need to extend our data sets with spam labels that identify the spam users or entries as such. For an adequate orientation we start by giving an overview of related work of combating spam in social bookmarking websites in Section 7.1. Section 7.2 then describes our two collections and how we obtained the spam labeling necessary for our experiments. It also describes the setup of our spam detection experiments, and contains an analysis of the spam problem for social bookmarking services (RQ 4). Then, in Section 7.3, we describe and evaluate a promising method of automatic spam detection using language models (RQ 4a). In Section 7.4 we investigate the influence that spammers and spam posts have on the recommendation quality (RQ 4b).

7.1 Related Work

Spam issues in social bookmarking services have received relatively little attention so far. Heymann et al. (2007) were the first to examine the relationship between spam and social bookmarking in detail. They classified the anti-spam strategies commonly in practice into three different categories: *prevention*, *demotion*, and *detection*.

Prevention-based approaches are aimed at making it difficult to contribute spam content to the social bookmarking system by restricting certain types of access through the interface (such as CAPTCHAs[2]) or through usage limits (such as post or tagging quota). The `nofollow` HTML attribute of hyperlinks can also serve as a spam deterrent, since it instructs search engines that a hyperlink should not influence the link target's ranking in the search engine's index, thereby removing the main motivation of spammers.

Demotion-based strategies focus on reducing the prominence and visibility of content likely to be spam. Rank-based methods, for instance, try to produce orderings of the system's

[2] A CAPTCHA is a challenge-response test used to prevent automated software from abusing online services. This is typically done by asking humans to perform a task that is difficult for a computer to perform properly, such as deciphering distorted characters (Von Ahn et al., 2008). The word 'CAPTCHA' is an acronym for 'Completely Automated Public Turing test to tell Computers and Humans Apart'.

content that are both more accurate and more resistant to spam (Heymann et al., 2007). A demotion-based strategy for combating spam is described by Heymann et al. (2007) and in more detail by Koutrika et al. (2007). The authors constructed a simplified model of tagging behavior in a social bookmarking system and compared different ranking methods for tag-based browsing. Moreover, they investigated the influence of various factors on these rankings, such as the proportion and behavior of spam users and tagging quota (Koutrika et al., 2007). More importantly, they found that approaches that take into account the agreement in tagging behavior between users are best able to decrease the influence of spam users for single-tag queries. Heymann et al. (2007) also found that restricting users to tag budgets—limiting the number of tags per users—decreased the influence of spammers, although at the cost of restraining the positive impact on the system due to good user activity.

Spam detection methods try to identify likely spam either manually or automatically, and then act upon this identification by either deleting the spam content or visibly marking it as such for the user (Heymann et al., 2007). Krause et al. (2008) describe, as far as we know, the only published effort of automatic spam detection with regard to social bookmarking websites. Krause et al. investigated the usefulness of different machine learning algorithms and features to automatically identify spam (Krause et al., 2008). They tested their algorithms on a data dump of the BibSonomy system and found that Support Vector Machines (SVM) and logistic regression performed best on the spam detection task. They examined the individual contribution of the different feature sets, and found that co-occurrence features—such as the number of co-occurrences of a user with spammers and non-spammers, and semantic features—contributed most to separating spammers from non-spammers.

In 2008, the Discovery Challenge workshop of the ECML/PKDD conference focused on two data mining tasks related to social bookmarking. One of these was detecting spam users in a social bookmarking system. So far, this has been the only TREC-like initiative focusing on the task of spam detection. With a total of 13 submissions, the majority of the participants' approaches used machine learning for the prediction task. Six out of the top eight approaches that beat the baseline used a large variety of different content-based and co-occurrence-based features combined with machine learning algorithms to separate the spammers from the genuine users (Hotho et al., 2008). The seventh of the top eight submissions used a graph-based algorithm for the detection task (Krestel and Chen, 2008). Our approach, described in Bogers and Van den Bosch (2008b) and in more detail in Bogers and Van den Bosch (2009a) (see also Section 7.3), finished fourth at the 2008 Discovery Challenge. Wetzker (2008) analyzed a large crawl of Delicious and briefly addressed the presence of spam in their data set. Contrary to the findings of Heymann et al. (2008a), they found a considerable spam presence on Delicious and reported that a large proportion of spam is created by a small number of users, suggesting automated posting routines for these accounts.

Casting our nets a bit wider than just social websites, we can also find a wealth of other anti-spam approaching in related fields such as blogs. Mishne et al. (2005) were among the first to address the problem of spam comments in blogs and used language model disagreement between the blog post itself, the comments, and any pages linked to from the comments to

identify possible spam comments. Their work inspired our approach to spam detection in social bookmarking. In 2006, the TREC Blog Track also paid attention to the problem of blog spam (Ounis et al., 2006) by examining the influence of spam posts on blog retrieval. Ounis et al. found that low performing systems were not more likely to retrieve more spam documents than high performing systems, and that spam posts did not appear to be a major hindrance to retrieval performance. In general, IR research on collections where a subset of the data has been manipulated for personal gain is known as *adversarial IR*, because of the opposing goals of the spammers and the search engine owners.

7.2 Methodology

To be able to answer RQ 4 and its two subquestions, we need access to data sets with manually identified spam objects. One of the two tasks in the 2008 Discovery Challenge was spam detection in a social bookmarking system. Here, we use the organizers' definition of the spam detection task to guide our experiments (Hotho et al., 2008). The goal of the spam detection task is to learn a model that predicts whether a user is a spammer or not. An added requirement is that the model should be able to accurately classify initial posts made by new users, in order to detect spammers as early as possible. The decision to identify spam in BibSonomy at the user level instead of at the post level implies that all of a spam user's posts are automatically labeled as spam. This decision was justified earlier in Krause et al. (2008) by the observation that users with malicious intent often attempt to hide their motivations by injecting non-spam posts[3]. In addition, Krause et al. also cite workload reduction as a reason for their decision to classify spam at the user level. The 2008 Discovery Challenge organizers report the following guidelines[4] for labeling users as spammers.

> *If we have the impression, that the user tries to advertise some of his web sites, tries to get backlinks to pages with a lot of google ads, or add links to offensive stuff etc. then we set all posts of this user 'private' and flag him as a spammer.*

In the experiments described in this chapter, we use the setup of the 2008 Discovery Challenge for our spam detection task and classify spam at the user level in both our BibSonomy and our CiteULike data set, to make for a fair comparison of our results. In the next subsections, we describe our data collection process (Subsection 7.2.1), our data representation format (Subsection 7.2.2), and how we evaluate the spam detection task (Subsection 7.2.3).

[3] Krause et al. were also the organizers of the 2008 Discovery Challenge, hence the same justification applies. Note, however, that the results reported in their 2008 paper were not achieved on the same data set as the one made available in the Discovery Challenge.

[4] As described in https://mail.cs.uni-kassel.de/pipermail/rsdc08/2008-May/000005.html

7.2.1 Data Collection

Automatic spam classification approaches typically demand a training or seed set to learn to predict spam characteristics (Heymann et al., 2007). So for us to be able to test our spam detection approach, we needed access to data sets with manually identified spam objects. We obtained such spam labels for data sets based on two social bookmarking websites: BibSonomy and CiteULike. The BibSonomy collection was pre-labeled as spam by the organizers of the 2008 Discovery Challenge. For CiteULike we annotated a sizable part of the collection ourselves. For Delicious, however, the considerably larger size of that data set and the stronger spam filtering policies (Heymann et al., 2008a) made it impractical to do the same. Hence, we excluded Delicious from this study. Table 7.1 provides statistics for the presence of spam in the CiteULike and BibSonomy collections. In both data sets spammers add two to three times as many tags to their posts on average than genuine users, thereby confirming what was already signaled in Krause et al. (2008): that tag count seems to be an informative feature for spam prediction. In the next two paragraphs we describe how we obtained our spam annotations and the specific characteristics of the two data sets.

Table 7.1: Spam statistics of the BibSonomy and CiteULike data sets. All CiteULike items were treated as references to scientific articles, since there is no clear-cut distinction between bookmarks and references on CiteULike. For BibSonomy, these are the counts of the training material combined with the official test set.

	BibSonomy	**CiteULike**
posts	2,102,509	224,987
bookmarks, spam	1,766,334	-
bookmarks, clean	177,546	-
references, spam	292	70,168
references, clean	158,335	154,819
users	38,920	5,200
spam	36,282	1,475
clean	2,638	3,725
average posts/user	54.0	43.3
spam	48.7	47.6
clean	127.3	41.6
tags	352,542	82,121
spam	310,812	43,751
clean	64,334	45,401
average tags/post	7.9	4.6
spam	8.9	7.7
clean	2.7	3.2

BibSonomy One of the data sets we used in our spam detection experiments was the BibSonomy data set, as introduced earlier in Subsection 3.2.2. It was supplemented with information about spam content as part of the 2008 ECML/PKDD Discovery Challenge, and contained flags that identified users as spammers or non-spammers. The Discovery Challenge organizers were able to collect data of more than 2,600 active users and more than 36,000 spammers by manually labeling users[5]. This reveals that the BibSonomy data

[5]This data set corresponds to the one described in Table 3.1.

set is strongly skewed towards spam users with almost 14 spam users for each genuine user. Table 7.1 also shows that spam users in BibSonomy clearly prefer to post bookmarks, whereas legitimate users tend to post more references.

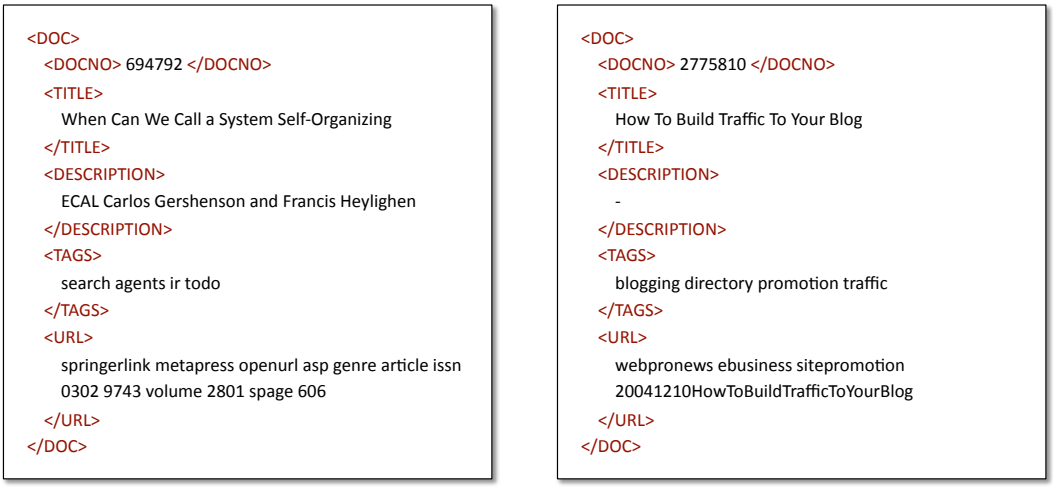
CiteULike Our self-crawled CiteULike data set did not include pre-labeled spam users or posts as the BibSonomy data set did. We therefore set out to collect our own spam labels for this data set. Here, we faced the same choice as the team organizing the Discovery Challenge: at which level of the folksonomy should we identify spam usage—users, items, tags, or individual posts? Our CiteULike collection contains over 1 million posts and over 800,000 items, and going through all of these was not practical. Judging all of the more than 232,000 tags was also infeasible, in part because it is simply not possible for many tags to unequivocally classify them as spam or non-spam. For instance, while many spam entries are tagged with the tag sex, there are also over 200 valid references on CiteULike that are tagged with sex. We therefore aimed to obtain an estimate of the pervasiveness of spam on CiteULike by identifying spam users. Judging all 25,375 users in the CiteULike data set would still be impractical, so we randomly selected 5,200 users (∼20%) from the data set and asked two annotators to judge these users on whether they were spammers or not. Each user was judged by only a single annotator to save time. Appendix A.2 goes into more detail about the spam annotation process, such as our annotation guidelines and the interface we used.

Of the 5,200 users in our CiteULike subset, 1,475 (or 28.1%) were spam users, which is a smaller proportion than in BibSonomy. The numbers in Table 7.1 are reported for this 20% sample of CiteULike users. An extrapolation of these proportions to the full CiteULike data set results in an estimated 7,200 spam users who posted references to CiteULike. To assess the accuracy of this estimation we may look at the problem from a different angle. As already remarked, certain spam references are removed quickly from the database by the CiteULike administrators, resulting in *404 Not Found* errors when crawling their reference pages. During metadata crawling of all 803,521 references in our November 7, 2007 data dump, about 26.5% of the references returned *404 Not Found* errors. A second round of re-crawling the metadata of these 213,129 missing references did not change this proportion. While spam removal is not necessarily the only reason for a *404 Not Found* error, we found that 18.7% of the 7,671 users that posted these 213,129 missing references were spam users identified in our annotation process, which is commensurate with the 20% sample we took. Furthermore, we found that 60,796 of the missing references (or 28.5%) belonged to the positively identified spam users. These estimates of 7,671 spam users (or 30.2%) and 213,129 spam references (or 26.5%) suggest that our extrapolation of spam presence on CiteULike is reliable.

7.2.2 Data Representation

After collecting the data we created a single representation format for all posts, capturing all relevant metadata in separate fields. As mentioned before, two types of resources can be posted to BibSonomy: bookmarks and BibTeX records, the latter with a magnitude more metadata available. Because there is no such clear distinction in our CiteULike data set, we

decided to treat BibTeX records and bookmarks the same and thus use the same format to represent both. We represented all resource metadata in an TREC-style SGML format using four fields: TITLE, DESCRIPTION, TAGS, and URL. URLs were pre-processed before they were used: punctuation was replaced by whitespace and common prefixes and suffixes like www, http://, and .com were removed. Figure 7.1 shows examples of clean and spam posts in our SGML representation. Whereas the clean post in Figure 7.1(a) is clearly recognizable as a regular scientific article, the emphasis on commercial incentives in Figure 7.1(b) signals the spam nature of that post.



(a) Clean post

(b) Spam post

Figure 7.1: Examples of clean and spam posts in our SGML representation.

A wide variety of metadata fields were available for the posts in the BibSonomy data set. For the bookmarks, the title information was taken from the BOOK_DESCRIPTION field in the MySQL dump, whereas the TITLE field was used for the BibTeX records. The DESCRIPTION field was filled with the BOOK_EXTENDED field for bookmarks, whereas the following fields were used for the BibTeX records: JOURNAL, PUBLISHER, ORGANIZATION, DESCRIPTION, ANNOTE, AUTHOR, EDITOR, BIBTEXABSTRACT, BOOKTITLE, HOWPUBLISHED, ADDRESS, SCHOOL, SERIES, and INSTITUTION. For both resource types all tags were added to the TAGS field. The URLs were extracted from the BOOK_URL and URL fields and pre-processed as described above.

Our post representations were significantly poorer for the CiteULike data set: since spam references were removed from the CiteULike website, we could not crawl the associated metadata of the spam references (cf. Subsection 7.2.1). Full metadata was available for the clean references, but using all metadata of the clean posts and only the tags of the spam posts would yield an unfair comparison. Any classifier would then simply learn a model where the presence or absence of metadata determines the predicted spam label. A system that predicts a post to be spam when it is missing metadata is unlikely to be very useful in any real-world situation. We therefore used only the tags for all CiteULike posts, clean and spam alike.

7.2.3 Evaluation

To evaluate our different approaches and optimized parameters, we divide each data set into a training set, a validation set, and a test set. Our models are trained on the training set and parameters were optimized on the validation set to prevent overfitting. For the BibSonomy data set, an official test set was supplied as part of the 2008 Discovery Challenge as well as training material, so we used this partitioning. We randomly selected 80% of the users from the training material for our training set and assigned the remaining 20% to our validation set. This gave us a training set of 25,372 users, a validation set of 6,343 users, and a test set of 7,205 users. For the CiteULike data set, we randomly selected 60% of all users for our training set, 20% for our validation set, and assigned the remaining 20% to our test set. This corresponded to 4,160 training users, and 520 validation set users, and 520 users in the CiteULike test set. For the final predictions on the test sets we used only the training sets we created to train our algorithm and generate the spam labeling predictions.

We evaluated our approaches on the validation and test sets using the standard measure of AUC (area under the ROC curve). The ROC curve plots the true positive rate of a classifier in the range $[0, 1]$ against the false positive rate in the domain $[0, 1]$. The diagonal in this graph represents the performance of a classifier that randomly guesses the output classes. Curves closer to the top left corner are better; a perfect ROC curve would go up horizontally from (0,0) to (0,1) and then vertically from (1,0) to (1,1). The AUC score represent the area under the ROC curve where 1.0 is the perfect score. We optimized k using AUC rather than on measures like the F-score, because AUC is less sensitive to class skew than F-score (Fawcett, 2004). This is especially relevant given that the data is rather skewed indeed, especially in the case of BibSonomy, with 12 spam users to every clean one.

7.3 Spam Detection for Social Bookmarking

Our approach to spam detection was inspired by the approach of Mishne et al. (2005) for dealing with spam in blog comments. It is based on the intuitive notion that spam users will use different language than 'legitimate' users when posting resources to a social bookmarking system. We detect new spam users in the system by first ranking all the old users in the system by the KL-divergence of the language models of their posted content and the language models of the new user's posts. We then look at the spam labels assigned to the most similar users in the system to predict a spam label for the new user. We propose using language models at two different levels of granularity: (1) at the level of separate posts, and (2) combined into user profiles. We will refer to these two variants of our approach as *user-level spam detection* and *post-level spam detection* respectively.

Both variants of our approach correspond to a k-nearest neighbor classifier with KL-divergence as its similarity metric. We will refer to our general approach as SPAMORY, because it is a memory-based spam detection algorithm. Each of the two variants consist of two steps: (1) calculating the similarity between users or posts, and (2) determining the final classification. In the next two subsections, we describe these two steps in more detail. Subsection 7.3.3 then presents our results, followed by a discussion in Subsection 7.3.4.

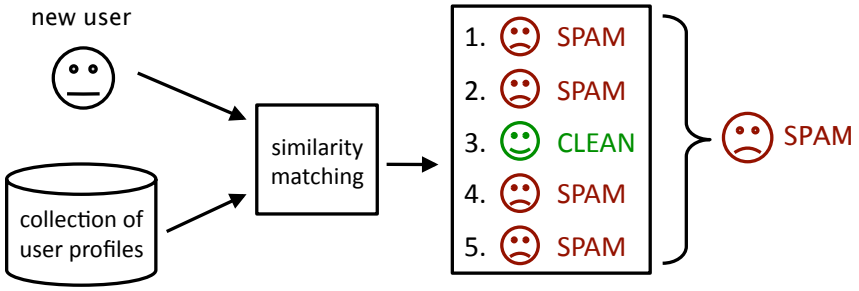
7.3.1 Language Models for Spam Detection

Language models (Jelinek, 1990) are a class of stochastic n-gram models, generally used to measure a degree of surprise in encountering a certain new span of text, given a training set of text. The core of most language models is a simple n-gram word prediction kernel that, based on a context of two or three previous words, generates a probability distribution of the next words to come. Strong agreement between the expected probabilities and actually occurring words (expressed in perplexity scores or divergence metrics) can be taken as indications that the new text comes from the same source as the original training text. Language models are an essential component in speech recognition (Jelinek, 1990) and statistical machine translation (Brown et al., 1990), and are also an important model in information retrieval (Ponte and Croft, 1998). In the latter context, separate language models are built for each document, and finding related documents to queries is transformed into ranking documents by the likelihood, estimated through their language model, that each of them generated the query. We do the same for SPAMORY.

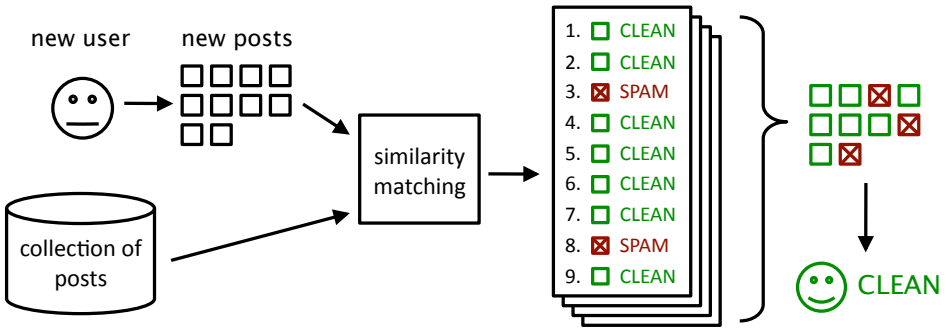
In generating our document language models, we have three different options for the granularity level of what span of text should be considered a document. At the most detailed level, we can construct a language model for each individual post, match these to the incoming posts, and use the known spam status of the best-matching posts already in the system to generate a prediction for the incoming posts or users. We can also take a higher-level perspective and collate all of a user's posts together to form merged documents that could be considered "user profiles", and generate language models of these individual user profiles. Incoming posts or users can then be matched against the language models of spammers and clean users to classify them as being more similar to one or the other category. A third option—at an even higher level of granularity—would be to only consider two language models: one of all spam posts and one of all clean posts. However, we believe this to be too coarse-grained for accurate prediction, so we did not pursue this further.

Figure 7.2 illustrates these two variants of SPAMORY. In the user-level variant depicted in Figure 7.2(a), the new user's profile—the concatenated collection of posts made by this user to the system—are matched against all existing user profiles. For each new user, this similarity matching process results in a list of the existing users, ranked by how similar their profiles are to the new user's profiles in terms of language use. The users that are most similar to the new user then determine the spam label. In this toy example, the majority of the top ranked users are spam users, so the classifier outputs the prediction 'Spam'.

In the post-level variant in Figure 7.2(b) the matching is done at the post level. Each of the new user's posts is matched separately against all the posts in the collection. For each new post, the matching process results in a list of the existing posts, ranked by how similar they are to the new post terms of language use. In the toy example in Figure 7.2(b) this results in ten different ranked lists, one for each new post. For each list separately, the best matching posts determine the spam label of the new post, resulting in a spam classification for each of the ten new posts. The final user spam label, the level at which we want our final predictions, is then generated by aggregating the predictions for the individual posts. In the toy example only three of the ten new posts were classified as spam, so the final user-level prediction is 'Clean'.



(a) User-level spam detection



(b) Post-level spam detection

Figure 7.2: Two variants of our spam detection approach

We used the Kullback-Leibler divergence metric to measure the similarity between the language models. The KL-divergence measures the difference between two probability distributions Θ_1, Θ_2 is

$$KL(\Theta_1||\Theta_2) = \sum_w p(w|\Theta_1) \log \frac{p(w|\Theta_1)}{p(w|\Theta_2)}, \tag{7.1}$$

where $p(w|\Theta_1)$ is the probability of observing the word w according to the model Θ_1 (Manning and Schütze, 1999; Mishne et al., 2005).

The Indri toolkit[6] implements different retrieval methods based on language modeling (Strohman et al., 2005). We used version 2.7 of this toolkit to perform our experiments and construct and compare the language models of the posts and user profiles. The language models we used are maximum likelihood estimates of the unigram occurrence probabilities. We used Jelinek-Mercer smoothing to smooth our language models, which interpolates the language model of a post or user profile with the language model of a background corpus (Jelinek, 1990)—in our case the training collection of posts or user profiles. We chose Jelinek-Mercer smoothing because it has been shown to work better for verbose queries than other smoothing methods such as Dirichlet smoothing (Zhai and Lafferty, 2004). Our user profiles and posts contain enough text to be considered verbose compared to standard IR query lengths. Preliminary experiments with Dirichlet smoothing confirmed this

[6]Available at `http://www.lemurproject.org`

for SPAMORY, as it was consistently outperformed by Jelinek-Mercer smoothing. While it is certainly possible to use other measures of document similarity, such as the cosine similarity, preliminary experiments with the Vector Space model and the cosine similarity metric consistently underperformed the language modeling approach. We therefore focused our efforts on language modeling and KL-divergence for user-user and post-post similarity.

We experimented with both the user-level approach and the post-level approach as illustrated in Figure 7.2. At the user-level, we compared the language models of the user profiles in our validation and test sets with the language models of the profiles in our training set. We then obtained a ranked list of the best-matching training users for each test user. We did the same at the post level by comparing the test post language models with the language models of the training posts. Here, ranked lists of best-matching posts were obtained for each test post. The similarity rankings based on the original similarity scores $sim_{original}$ were normalized into $[0, 1]$ using the maximum and minimum similarity scores sim_{max} and sim_{min} by the formula from Lee (1997):

$$sim_{norm} = \frac{sim_{original} - sim_{min}}{sim_{max} - sim_{min}}. \tag{7.2}$$

These similarity rankings were normalized, and used as input for the spam classification step. In our BibSonomy data set we have four different metadata fields available to generate the language models of the posts and user profiles in our training collection: title, description, tags, and tokenized URL. In addition to these 'complete' runs with all fields, we also ran experiments where we only used the information from the four fields separately. An example would be to use only the tags from the training users and the test users. This resulted in five different runs for BibSonomy. For CiteULike we only had the tags available, so we performed only one run here. We experimented with different combinations of metadata fields and refer the reader to Bogers and Van den Bosch (2009a) for more details.

7.3.2 Spam Classification

After we generated the language models for all posts and user profiles, we obtained the normalized rankings of all training documents, relative to each test post or user profile. For each of the best-matching training documents, we used the manually assigned spam labels ('0' or '1') to generate a single spam score for the new user. The simplest method of calculating such a score would be to output the spam label of the top-matching document. A more elegant option would be to take the most common spam label among the top k hits. We settled on calculating a weighted average of the similarity scores multiplied by the spam labels, as preliminary experiments showed that this combination outperformed the other options. Spam scores $score(u_i)$ for a user u_i were calculated according to the following equation:

$$score(u_i) = \frac{\sum_{r=1, r \neq i}^{k} sim(u_i, u_r) \cdot label(u_r)}{k}, \tag{7.3}$$

where for the top k matching users u_r from ranks 1 to k the similarity score $sim(u_i, u_r)$ between the user in question u_i and the matching user u_r is multiplied by the spam label $label(u_r)$ of that matching user. The total weighted scores are divided by the number of matches k, yielding a weighted average score for u_i.

For post-level classification, this meant that we obtained these weighted average spam scores on a per-incoming-post basis. To arrive at user-level spam scores, we then matched each incoming post to a user and calculated the average per-post score for each user. In the rare case that no matching documents could be retrieved, we resorted to assigning a default label of no spam ('0'). In the BibSonomy validation set, for instance, this only occurred for 0.7% of users. Our default classification for these users was 'clean' because 84.2% of these users in the BibSonomy validation set were legitimate users.

After implementation of all these techniques, one question still remains: how many of the top matching results should be used to predict the spam score? Here, our approach is similar to a k-nearest neighbor classifier, where the number of best-matching neighbors k determines the prediction quality. Using too many neighbors might smooth the pool from which to draw the predictions too much in the direction of the majority class, while not considering sufficient neighbors might result in basing too many decisions on accidental similarities. We optimized the optimal value for k for all of the variants separately on the AUC scores on the validation set. These optimal values of k were then used to calculate the final scores on the test sets.

7.3.3 Results

Table 7.2 lists the outcomes of the two different variants of our SPAMORY approach on the two collections. Since we optimized on the validation sets, we mainly focus on the test set scores to draw our conclusions. The best performing approach on BibSonomy, at an AUC score of 0.9661, is spam detection at the user level, using all available metadata fields. The best post-level run on BibSonomy also used all of the data for all of the posts, and achieves a score of 0.9536. On the CiteULike data set, the best performance at the user level and post level yields AUC scores of 0.9240 and 0.9079, respectively. This seems to suggest that our approach generalizes well to other data sets and social bookmarking systems. We observe that in general, using the language models constructed at the user level outperforms using the post-level language models. This is also visible in Figure 7.3, which shows the ROC curves for the best user-level and post-level runs for each collection.

An interesting difference between the validation set and the test set is that using only the tags to construct the language models yields the best performance on the validation set, whereas performance using only tags drops markedly on the test set. Using all available metadata fields results in a considerably more stable performance across both BibSonomy evaluation sets. It should therefore be considered as the preferred variant.

In contrast, we observed that most of the other runs using different combinations of metadata fields tend to perform visibly better on either the validation or the test set. A third interesting observation is the difference in the optimal size of the neighborhood k used to

Table 7.2: Results of our two spam detection variants on the BibSonomy and CiteULike data sets. Scores reported are AUC, with the best scores for each set of collection runs printed in bold. The optimal neighborhood size k is listed for each user-level and post-level runs. For the same set of runs, the same value of k was used in both the validation and the test set.

Collection	Fields	User level			Post level		
		Validation	Test	k	Validation	Test	k
BibSonomy	All fields	0.9682	**0.9661**	235	0.9571	**0.9536**	50
	TITLE	0.9290	0.9450	150	0.9055	0.9287	45
	DESCRIPTION	0.9055	0.9452	100	0.8802	0.9371	100
	TAGS	**0.9724**	0.9073	110	**0.9614**	0.9088	60
	URL	0.8785	0.8523	35	0.8489	0.8301	8
CiteULike	TAGS	**0.9329**	**0.9240**	5	**0.9262**	**0.9079**	5

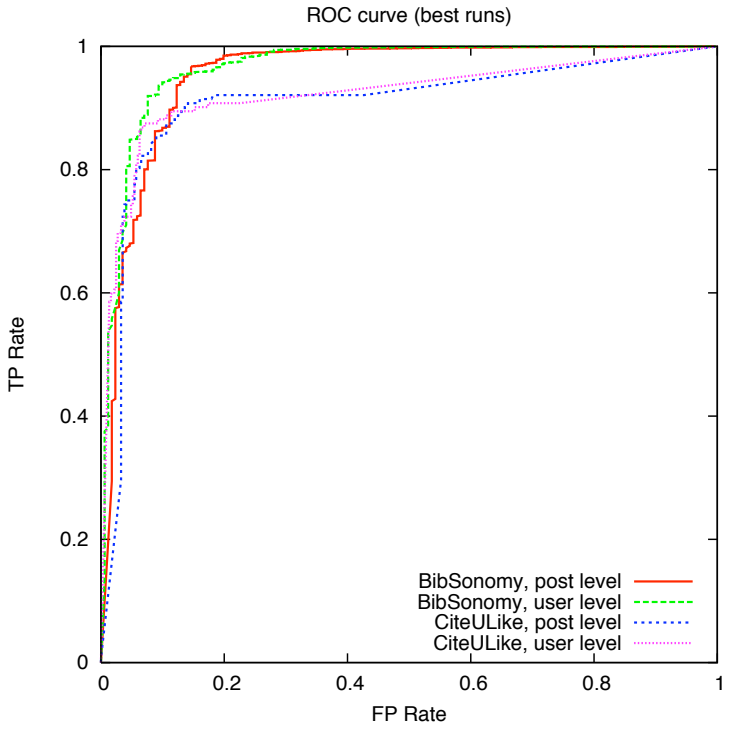


Figure 7.3: ROC curves of the best-performing user-level and post-level approaches for both collections. Curves closer to the top left corner are better; a perfect ROC curve would go up horizontally from (0,0) to (0,1) and then vertically from (1,0) to (1,1).

predict the spam labels. In almost all cases, the post-level variants require a smaller k than at the user level. The optimal neighborhood size for CiteULike is the same for both the user-level and the post-level variants, and surprisingly smaller than for BibSonomy.

Comparison to Other Approaches While we have not compared SPAMORY to standard spam detection approaches such as SVMs or Naive Bayes directly, we can use the outcome of the spam detection task at the 2008 Discovery Challenge to measure SPAMORY against such approaches. Our updated version of SPAMORY would have achieved the third place with an AUC score of 0.9661, surpassed only by a ridge regression approach with extensive feature preparation and selection, and a SVM approach to spam detection (Hotho et al., 2008). Four

of the top eight approaches used SVM; the other three SVM approaches performed worse than the approach described in this paper. One participant used Naive Bayes learning with extensive feature selection, achieving slightly lower scores than our approach. Finally, one of the participants compared five different machine learning methods on the spam detection task, with none of the five outperforming our kNN-based approach, suggesting that SPAMORY is competitive with other machine learning methods.

A particular advantage of our SPAMORY approach is that it could be implemented with limited effort on top of an existing social bookmarking search engine. After retrieving relevant results for a user query, the top k matching results can then be used to generate the spam classification, requiring only a lookup of predetermined spam labels. Other machine learning approaches would require a more elaborate implementation without being able to leverage the existing search infrastructure of a social bookmarking website.

7.3.4 Discussion and Conclusions

In this chapter we presented SPAMORY, an adversarial information retrieval approach employing language modeling to detect spam in social bookmarking websites. We started by using language models to identify the best-matching posts or user profiles for incoming users and posts. We then looked at the spam status of those best-matching neighbors and used them to guide our spam classification. Our results indicate that our language modeling approach to spam detection in social bookmarking systems is promising, yielding AUC scores of 0.953 and 0.966 on spam user detection. This confirms the findings of Mishne et al. (2005), who applied a similar two-stage process using language modeling to detecting blog spam, albeit on a smaller scale. With our approach and experimental setup we have improved upon the work described in Mishne et al. (2005) in two ways. One improvement is in scale: the data set we used, is several orders of magnitude larger than the one used by Mishne et al., with 43000+ users and 2,300,000+ posts divided over two different data sets, compared to their 50 blog posts and 1024 comments (Mishne et al., 2005). Since SPAMORY was inspired by their approach, our results serve to confirm that using language model similarity is a good approach for spam detection on a much larger scale as well. A second improvement can be found in the evaluation measure used. In the data set used by Mishne et al. 68% of the content was spam and in our two data sets these proportions were around 93% and 28%. When evaluating on such skewed data sets it is much better to use a measure such as AUC than to use the simple accuracy metric used by Mishne et al. (Fawcett, 2004).

Language Model Granularity We experimented with using language models at two different levels of granularity and found that matching at the user level and using all of the available metadata gave the best results. In general, matching at the user level resulted in better performance than matching at the post level for both BibSonomy and CiteULike.

This difference can be partly explained by the fact that the spam labels in both data sets were judged and assigned at the user level, because this is the desired level of the end application. Even if a spam user posts some 'genuine' posts, the entire content of the spam user should be deleted on grounds of the adversarial intentions behind them. Yet, those

'genuine' posts of spam users are then automatically flagged as spam, thereby introducing more noise for the post-level classification than for the user-level classification. Depending on whether the user's intentions were genuine or not, the same article or website could be posted multiple times, sometimes labeled as clean content, other times labeled as spam content. This could potentially confuse the post-level classifier, but not the user-level classifier. Early classification of spam users at their earliest posts can therefore be expected to be less accurate than the reported 0.95–0.96 range; post-level AUC scores suggest this accuracy would be closer to 0.91–0.95.

A second likely explanation for the better performance of the user-level approach is metadata sparseness at the post level. A post-level approach is more likely to suffer from incoming posts with sparse or missing metadata. For instance, although 99.95% of all posts in the BibSonomy data set have valid tags[7], this also means that it is possible for incoming posts to have no tags. Without any tags as metadata or sparse metadata in the other fields, our post-level SPAMORY variant cannot find any matching posts in the system. At the user level, this is much less likely to happen: only 0.009% of all users never assign any tags. Aggregating all metadata of a user's posts can yield sufficient metadata to base reliable predictions on, whereas the post-level approach can be affected by this to a greater extent. Missing tags might also be a reason for the fact that performance on CiteULike is slightly lower than performance on BibSonomy.

Neighborhood Size When looking at the optimal neighborhood sizes k for BibSonomy, we see that in almost all cases the post-level approaches required a smaller k than at the user level. We believe that this is because the presence of multiple topics in user profiles. Individual posts are usually about a single topic, whereas a user profile is composed of all of that user's posts, which are likely to be about multiple topics of interest. This makes finding the related posts to an individual post easier, in the sense that it requires fewer nearest neighbors to arrive at a prediction. At the user level, however, different parts of a user's profile might match up with different users already in the system, thus requiring more nearest neighbors to arrive at a reliable prediction. For CiteULike the optimal k values are the same for both the post level and the user level, but we have no plausible explanation for this.

Future Work No spam detection approach can be expected to remain successful without adapting to the changing behavior of the spammers. One way spammers could circumvent our SPAMORY approach would be by generating metadata with a similar language model to the clean posts in the system. This way, spammers can make it more complicated for our approach to distinguish between themselves and genuine users. However, this also makes it more difficult for the spammers themselves: it is very hard for a spammer to post resources to a social bookmarking system that will be both similar to existing posts and to the language of the spam entry (Mishne et al., 2005). In addition, such behavior could be countered by extending our method to include the language models of the pages and documents behind

[7]Valid meaning with a tag other than system:unfiled, the default tag that is assigned by the system when no tags were added by the user.

the bookmarks[8]. In the case of sparse metadata, this might be able to boost performance of the spam detection algorithm. Extending SPAMORY in such a way is one of the possible avenues for future work. A second option would be to include extra features such as the PageRank scores (Page et al., 1998) of the bookmarked pages, and see whether pages with low PageRank are more predictive of spam status than others.

7.4 The Influence of Spam on Recommendation

In the previous section, we described a successful approach to the problem of detecting spam in social bookmarking systems. Spam content is a nuisance to all parties involved but the spammers themselves. For users, spam pollutes the result lists when searching or browsing a social bookmarking website, whereas from the system administrators' standpoint it wastes both computing power and storage capacity. Adequate spam detection and removal is therefore an important component of any mature social bookmarking service. Removing spam is also a boon to the performance of a recommender system. Recommending spam lowers the trust users place in the system, and can impair the efficiency and effectiveness of recommendation algorithms such as CF. Removing spam users and items means they do not have to be taken into account anymore when calculating the similarities between users and items, severely cutting down on computation time. The question we wish to answer in this section is how much recommendation effectiveness is weakened by the presence of spam. In other words, how much is recommendation performance degraded by spam content, and how often does spam content show up in the results? Our experiments here should be regarded as a case study and a first look into the problem of spam for item recommendation on social bookmarking websites.

7.4.1 Related Work

Most of the related work on spam and recommendation has focused on the creation of more robust recommendation algorithms that are less affected by spam. Spam attacks on recommender systems are commonly referred to as *shilling attacks* (Lam and Riedl, 2004). In such attacks, a malicious user creates multiple fake identities in the recommender system. Using these fake profiles, coordinated sets of spam user profiles with ratings are injected in such a manner that they affect the recommendations made by the system to the advantage of the spammer. Individually, each user profile will not affect the recommendations, but, when combined, the profiles are designed to boost certain items in favor of others. Two types of shilling attacks are distinguished: *push attacks*, that attempt to promote a certain item in the recommendations, and *nuke attacks*, that are aimed at downgrading specific items (O'Mahony et al., 2004). Several robust recommendation algorithms have been proposed in recent years (O'Mahony et al., 2004; Mobasher et al., 2006; Resnick and Sami, 2007; Bryan et al., 2008; Mehta, 2008), and they have shown that, in general, model-based algorithms

[8] Although it should be noted that it is far from trivial to obtain the full text of all the source documents linked to by the BibSonomy and CiteULike posts. Moreover, we suspect that incorporating language models from all externally linked Web pages and documents would slow down a real-time spam filtering system to an undesirable degree.

are more robust to shilling attacks than memory-based algorithms. In this section we are not interested in designing more robust recommendation algorithms, but in doing exploratory work in quantifying the effects of spam on some of our own algorithms. We leave a more extensive investigation and evaluation for future work.

7.4.2 Experimental Setup

Below we test the influence of spam content on recommendation using our BibSonomy collection. In Chapter 3 we described how we created our two data sets BibBoo and BibArt using this collection. For the experiments described in this section, we extend these data sets with all the spam content that we filtered out for our main experiments. To make for a fair comparison, we apply the same filtering to these data sets as we did to BibBoo and BibArt: we only retain users with 20 items or more, and only keep items that occur in at least 2 user profiles. The BibArt part of our BibSonomy collection hardly contains any spam as we already showed in Table 7.1. After this filtering took place there was virtually no spam content left, which would render a comparison using the BibArt data set useless. Furthermore, we do not use the CiteULike data set for the experiments described in this section. This is for two reasons. First, we only have spam labels available for a 20% subset of our CiteULike data set. Experiments using such a small data set would not be representative of a real world situation. Second, we do not have metadata available for the spam entries in our CiteULike data set, making a comparison impossible for the metadata-based approaches. This leaves us with the BibBoo data set to investigate the influence of spam on recommendation. Table 7.3 offers a basic description of the new, spam-extended data set compared to the old, spam-free one.

Table 7.3: Basic statistics of our spam-extended BibSonomy bookmarks data set compared to the original spam-free data set.

	Spam-free data set	**Spam-extended data set**
# users	192	4,914
# items	11,165	177,139
# tags	13,233	95,796
# posts	29,096	634,129
avg # items per user	151.5	129.0
avg # users per item	2.6	3.5
avg # tags per user	203.3	325.2
avg # users per tag	2.9	16.7
avg # tags per item	8.4	13.3
avg # items per tag	7.1	24.6

The numbers in Table 7.3 confirm what we already saw in Table 7.1: spam users add significantly more tags to their items than genuine users do. Moreover, in the spam data set, users have a lower average profile size, but a higher average number of users per item. This strongly suggests that spammers enter multiple, near-duplicate user profiles that all refer to the same spam content, as was already suggested by O'Mahony et al. (2004). This might indicate that spam users tend to be very similar in their behavior to each other, but perhaps not to the genuine users.

Which algorithms do we use to test to determine the influence that spam content has on item recommendation? We selected the approaches from Chapters 4 and 5 that performed best on the BibBoo data set: (1) item-based filtering using tag overlap similarity, and (2) profile-centric content-based filtering. We selected IT-JACCARD-SIM as our similarity metric for the item-based filtering algorithm, and optimized the neighborhood size for the spam extended data sets using our 10-fold cross-validation setup. For the profile-centric approach we used all of the intrinsic metadata fields for our user and item representations. Using the spam-extended version of our BibSonomy bookmarks data set, we considered all users and items for calculating the similarities. We only evaluated, however, on the users present in the clean data sets, because we are not interested in how well we can recommend content to spam users. We evaluated on MAP and on the average percentage of spam content in the top 1, top 10, top 25, top 50, top 100, and top 1000 results. We derived a list of spam content by assuming all the items not present in our original, spam-free data set were spam. We know this is not a perfect assumption, as spammers tend to include 'genuine' posts to throw off spam detection efforts. If these 'genuine' items were not in the original spam-free data set, then they were misclassified as spam. However, we do not believe this to be a big problem.

7.4.3 Results and Analysis

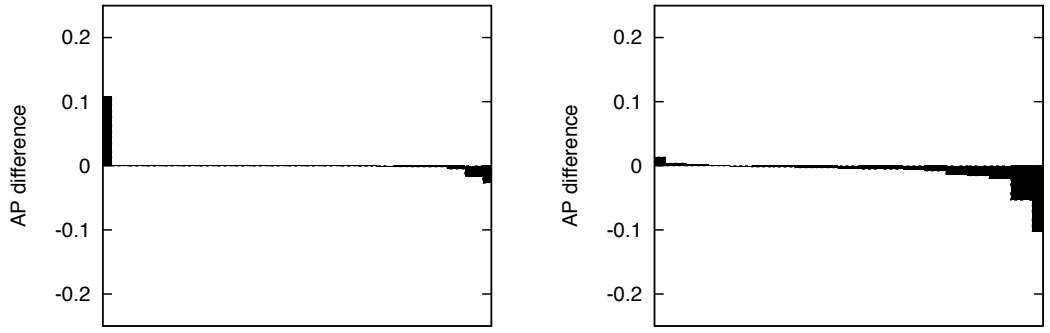
Table 7.4 contains the results of our two selected algorithms on the BibBoo data set with and without spam. What we see is that the content-based approach is affected most by the presence of spam: the MAP score on the spam-extended data set decreases by almost 30% from 0.0402 to 0.0283, although the difference is not statistically significant ($p = 0.060$). Surprisingly, the item-based filtering run using tag overlap achieves a slightly higher MAP score on the spam-extended data set. The increase from 0.0370 to 0.0397 is not statistically significant.

Table 7.4: The influence of spam on recommendation performance on our original BibSonomy bookmarks data set and the spam-extended version. Reported are MAP scores and the percentage of spam items at different cutoffs in the ranking.

	Tag overlap	**Content-based**
MAP on spam-free data set	0.0370	0.0402
MAP on spam-extended data set	0.0397	0.0283
Top 1 spam %	10.5%	0.0%
Top 10 spam %	15.3%	4.2%
Top 25 spam %	14.9%	5.9%
Top 50 spam %	15.8%	6.7%
Top 100 spam %	13.6%	7.0%
Top 1000 spam %	4.0%	18.2%

Figure 7.4 shows the per-user differences in Average Precision (AP) on the spam-extended and spam-free data sets for both approaches. It can be observed from Figure 7.4(a) that the increase in MAP for the tag overlap approach is due to the large improvement for a single user. For all other users, performance remains the same or decreases slightly. This leads us to believe that the increase in MAP is coincidental and not structural. In contrast, Figure

7.4(b) shows a consistent, negative effect of spam on the AP scores of the majority of test users.



(a) Item-based CF with tag overlap

(b) Profile-centric approach

Figure 7.4: User level differences in AP scores between the runs on the spam-extended and spam-free data sets. Figure 7.4(a) shows the score differences for the item-based CF approach with tag overlap similarity; Figure 7.4(b) shows the differences for the profile-centric approach.

If we look at the actual spam presence in the rankings of the clean and spam-extended data sets in Table 7.4, we can see some remarkable differences between the tag-based and content-based runs. While the content-based run contains more spam items in its rankings overall, we see that this spam mostly concentrated in the lower end of the rankings, after the top 100 results. In contrast, for the tag overlap run, most of the spam content is present in the top 100 of each user's list of recommendations. This percentage is fairly constant at around 15%, which means the spam items are evenly distributed over the top 100 items. In 10% of the cases, even the first item recommended by the tag overlap approach is spam; this never happens for the content-based approach.

The tag overlap approach is not harmed by spam in the data, while the content-based approach is, even though the item rankings of the former are in fact polluted with more spam in the top 100. This difference is surprising, since we could reasonably expect the approach with the biggest proportion of spam in the ranking to perform worst, as well as other approaches where the spam content is located at the top end of the rankings. The fact that this does not happen shows that the presence of spam does not change the ranking of the relevant items by the item-based filtering approach. In the content-based approach, relevant items are actually ranked worse when spam is present, suggesting a precision-decreasing effect. Using the same analysis as described in Subsection 6.4.1, we are able to confirm this. For both spam-extended runs, recall is down as they retrieve less of the relevant items. However, the items are ranked better by the spam-extended tag overlap approach as we see a precision-enhancing effect of a 6.3% increase in MAP compared to the run on the spam-free data set. Moreover, we also see a recall-decreasing effect of -9.7%. For the content-based approach, we instead see a precision-related -29.9% decrease in MAP score and a recall-related decrease of -21.1%.

Let us attempt to find out what is happening to the rankings generated by the content-based approach that causes this decrease in performance. Our content-based recommendation algorithm is based on a standard language modeling approach to IR, where the user profile representation corresponds to the query and the item representations correspond to the documents. In language modeling, the goal is to estimate the likelihood of the query being generated by a documents, and then ranking all documents by those estimated likelihoods. We use the Jelinek-Mercer smoothing method in our experiments, which involves linear interpolation between a document model and a collection model (Hiemstra, 1998). For each query term—i.e., term in the user profile—we calculate the probability of the term being generated from the document model θ_d according to the equation

$$p(t|\theta_d) = \lambda \cdot p(t|d) + (1 - \lambda) \cdot p(t), \tag{7.4}$$

where $p(t|d)$ represents the term probability according to the document model and $p(t)$ represents the term probability according to the background collection model. The weighting parameter λ was set to 0.9 in our experiments. When we add large amounts of spam to a data set as we did for BibBoo, the queries and the document models for a particular query-document pair—or in our case user-item pair—do not change. What can and does change, however, is the collection model and the associated term probabilities $p(t)$. The probability of a term given the collection model is calculated as the total number of occurrences of t in the collection $\sum_d n(t,d)$ divided by the total number of terms in the collection $\sum_d n(d)$, as in

$$p(t) = \frac{\sum_d n(t,d)}{\sum_d n(d)}, \tag{7.5}$$

where $n(t,d)$ is the number of occurrences of term t in document d and $n(d)$ is the total number of terms in document d. We see that the ranking of the relevant items decreases, which could mean one of two things. One possibility is that $p(t)$ decreases for the terms also occurring in the user's profile. This can happen when the spam items introduce more terms into the collection that are not in the user profile. When this happens, the total number of terms in the collection $\sum_d n(d)$ grows faster than the number of occurrences of a specific term in the collection $\sum_d n(t,d)$. The other possible reason for the rankings drop of relevant items is that other non-relevant, non-spam items are ranked higher and repress the relevant items. This could happen when certain user profile terms that were not matched against the relevant items in the spam-free data set suddenly occur much more due to the spam content and become more important for ranking the other non-relevant items. The probability $p(t)$ for these terms could increase either through a decrease in collection size—which is impossible here—or through an increase in the occurrence of term t in the collection. In the latter case, the spam items contain many occurrences of these previously 'unused' terms and increase the importance of the background model in ranking the items. We believe it is a combination of these two factors that causes the performance drop.

In contrast, the item-based filtering approach with tag overlap is not affected because more items are added to the system. For this approach to be affected, the newly injected items

also have to directly match the items of a specific user to influence the recommendations for that user. Since the MAP score of the tag overlap approach is not affected, spammers do not appear to directly target the system in this way. The item-based CF algorithm has also been shown in the past to be fairly robust against spam (Lam and Riedl, 2004). The end goal of spammers, however, is not per se to ruin the ranking: it is to manipulate it in such a way that their own content can be found at the top of the results lists. The tag overlap approach does not seem to be better at this than the content-based approach, even though the previously added items are retrieved better by the former approach when spam is present in the system. When all is said and done, the most important vote here is cast by the users themselves, who do not appreciate the presence of spam in their list of recommendations, even when it is alternated with relevant recommendations. We believe this to be the best reason for social bookmarking systems to filter out spam content.

7.5 Chapter Conclusions and Answer to RQ 4

In this chapter we have attempted to determine what the influence of spam content is on social bookmarking systems, corresponding to our fourth research question and its two subquestions.

RQ 4 How big of a problem is spam for social bookmarking services?

RQ 4a Can we automatically detect spam content?

RQ 4b What influence does spam have on the recommendation performance?

We argued that spam is as much of a problem for social bookmarking as it is for other tasks such as e-mail and search. To quantify the problem, we examined two of our collections, CiteULike and BibSonomy in more detail, and found that these data sets contain large amounts of spam, ranging from 30% to 93% of all users marked as spammers (RQ 4). We then proposed SPAMORY, a k-NN approach to spam detection that builds representations of the user and item metadata in a system, and uses the KL divergence as its similarity metric. We implemented SPAMORY at two different levels of granularity: directly matching user profiles and matching item profiles. We found that using all available metadata at the user level provides the best performance, and that our spam detection approach works well on both data sets (RQ 4a). Finally, we took a closer look at the influence of spam on recommendation performance, and extended our BibSonomy bookmarks data set with all spam content. We tested a CF and a content-based approach, and found that spam has a negative effect on recommendation. The content-based approach was affected most by the spam presence, but all result lists were unacceptably polluted with spam items, proving the necessity of adequate spam detection techniques (RQ 4b).

m

Chapter 8

Duplicates

Another growing pain that social bookmarking systems face, and the second we focus on, is the presence of duplicate items in the system. With many items being bookmarked by different users, the ideal situation would be that, for each new post, the system could recognize if that resource has already been posted to the system. Although each post is personal to a user and should be stored separately along with the personal tags and metadata for that user-item combination, the common resource targeted in all those posts should only be stored once in the system. While most services will attempt to match identical resources—with varying degrees of sophistication—no matching system can be perfect. Compared to spam, this is obviously less an issue of malicious intent, and more one of carelessness or lack of awareness. However, their presence can still have consequences for the social bookmarking service. Users might not be able to find all related users based on the articles they share if duplicates create a chasm between different user groups. Erroneously storing the same resource multiple times could diminish the strength of the social bookmarking system and deflate popularity counts, as well as increase storage and processing requirements. It might also affect item recommendation: two versions of the same article might be recommended in the same session, and perhaps it is not possible to locate all interesting articles when article counts are diluted and fragmented. In this chapter, we focus on this problem: how can we detect duplicates automatically and what effect do they have on recommendation? In other words, how important is it to have a good de-duplication policy in a social bookmarking system? This leads us to our fifth research question.

RQ 5 How big a problem is the entry of duplicate content for social bookmarking services?

The problem of duplicates exists, to varying degrees, in all three social bookmarking services: CiteULike, BibSonomy, and Delicious. However, we will only examine it for CiteULike in this chapter. Jäschke et al. (2007a) mention that BibSonomy already uses hashing at the posting phase to make users attentive of the fact they might be adding a resource already present in the system. The BibSonomy data set is therefore not very suited for our experiments. While both Delicious and CiteULike could have been used for our investigations in

this chapter, we chose to focus only on CiteULike because of its smaller size, allowing for the manual annotation of a reasonable portion of the data. Our fifth research question gives rise to two, more specific subquestions.

RQ 5a Can we construct an algorithm for automatic duplicate detection?

RQ 5b What influence do duplicates have on recommendation performance?

In the next section we describe how duplicates are currently handled in CiteULike, followed by Section 8.2 where we discuss the most appropriate related work in de-duplication. In Section 8.3, we describe our approach for detecting duplicates in the CiteULike collection, from creating an appropriate test set to doing the actual de-duplication and its results (RQ 5 and RQ 5a). Section 8.4 then describes our investigations into the influence duplicates can have on recommendation (RQ 5b).

8.1 Duplicates in CiteULike

CiteULike users have many different options for adding articles to their own CiteULike profile. The easiest option is by simply clicking on the 'Copy' link when viewing the dedicated page of an interesting article. However, the copy option is not the only method for adding articles to a personal profile, and many users do not search CiteULike to see if the article they want to add is already present. In the 30 days leading up to the Christmas holidays of 2008, CiteULike reported to have received 116,477 posts, of which only 6,189 posts (or 5.3%) were copies made directly from other users' libraries (Fenner, 2009). Judging by the many import-related tags in the CiteULike data set[1], a popular way of adding articles to one's library is by importing an entire BibTeX or EndNote library at once. If these entries do not already contain universal document identifiers like DOI numbers, then no steps are taken by CiteULike to match these imports to existing items in the database (Chris Hall, Citeulike, personal communication, May 2008).

We expect that, because of these two factors, a sizable part of the articles posted to CiteULike are duplicates of items already in the database. Since popular articles are added more often, there is a bigger chance that something can go wrong when users enter the item metadata into the system. It is reasonable to assume that there is a fixed probability that spelling errors or missing DOIs cause a duplicate article to be spawned. The number of duplicates for a given article is therefore likely to be commensurate with the popularity of the first instance of that article posted to CiteULike. We examine in Section 8.3 whether this relationship is indeed present.

In terms of popularity of the duplicate items themselves, one would expect that the duplicate versions of an article are rare themselves. Because the percentage of articles that are copied directly to a user's profile via the 'Copy' link is relatively small, duplicate versions are not

[1] We manually identified 57 different import-related tags, such as bibtex-import or endnote_import. These tags comprise up to 3.8% of all assigned tags in the November 2, 2007 data set. Most people add 'real' tags themselves after importing, so the number of items added by batch import is likely to be even higher.

likely to propagate this way. Nevertheless, this is not always the case: the original version of the article "Referral Web: Combining Social Networks and Collaborative Filtering" (Kautz et al., 1997) has been added by 40 users, whereas an exact duplicate record of that paper that was added later to CiteULike has been added by 31 other users. In this case, the duplicate version is nearly as popular as the original version of the same paper. If we assume that this article spawned other, less frequent duplicate versions as well, this means that the article is twice as popular on CiteULike than one would expect from simply observing the frequency of the original version of the article. In general, however, we expect that all duplicate versions on CiteULike together show a Zipfian frequency distribution with a long tail of unpopular duplicate versions and a handful of duplicates that have become relatively popular.

8.2 Related Work

Any recommender, retrieval, or information access system that relies on user-generated content or metadata is susceptible to the problem of duplicate or near-duplicate content. In dealing with such duplicate content, system designers have a choice of when to take action: at the moment duplicate content is introduced into the system, at the end when the results of some requested action are presented to the user, or somewhere in between. Deferring de-duplication to the end is a *lazy* approach, whereas the other two options can be classified as *eager* approaches to de-duplication.

Lazy Deduplication The lazy approach to de-duplication is most common in the literature on recommender systems. Here, the problem of removing duplicate content is deferred to after the recommendations have been generated, either by grouping duplicate items together or by removing them from the final list of recommendations presented to the user. For instance, search engines for research papers such as Google Scholar[2] attempt to group all duplicate versions of the same reference together in their search results. Online stores such as Amazon[3] form another example as they try to group together the different formats of the same item— such as Blu-ray and DVD versions of the same movie—in their recommendation lists. In the majority of the literature on recommender systems, de-duplication is mentioned merely in passing. Most papers mention having a de-duplication module that performs post-processing on the final recommendation lists, but do not go into any more detail about this (Glance et al., 2001; Haase et al., 2004; Horozov et al., 2006). The way the de-duplication problem is described in these papers suggests that identifying duplicate content is often dealt with as an afterthought in the design of such complex systems.

Eager Deduplication In contrast, taking an eager approach means the de-duplication process takes place before the actual recommendation, retrieval, or aggregation operations are executed. Eager de-duplication has several advantages over its lazy counterpart, such as obtaining a cleaner and more compact data set, reducing storage and processing requirements. In the eager scenario each system module can operate on the same 'pure' data set,

[2] http://scholar.google.com/
[3] http://www.amazon.com/

whereas the lazy situation would require separate de-duplication post-processing steps to be run on their separate outputs. Arguably the optimal stage to detect duplicates is when they are entered into the system if there is a way to involve the user in this. Presenting the user with a list of similar content already into the system, could prevent the user from polluting the database with identical information. For practical reasons, however, it might not always be possible to involve the user in this process—users might choose to upload a large batch of their publications that they are not willing to de-duplicate in a single pass. Identifying all duplicates in a (periodic) batch process after data entry but before recommendation will often be the most practical option. As we crawled our data sets from the Web, we cannot elicit the help of users directly, which leads us to resort to a batch approach to de-duplication as well.

We know of no related work on eager de-duplication in the field of recommender systems. In the past 50 years, however, many different research fields have investigated the problem of (near-)duplicate content under various monikers, including record linkage, data integration, database hardening, name matching, and entity resolution (Elmagarmid et al., 2007).

Deduplication on the Web Finding near-duplicate content among Web pages has been an active research topic in the IR community for more than a decade. It is aimed at reducing the index sizes of Web search engines, and prevalent approaches operate on the full text of billions of Web pages. We refer the reader to Henzinger (2006) for a comprehensive comparison and overview of the subject. The techniques described in Henzinger (2006) are optimized for finding (near-)duplicates of Web pages using the full text. In our approach we focus on de-duplication using techniques from the record linkage field, as these are optimized on working with metadata content from database fields[4].

Deduplication in Databases In our CiteULike data set we wish to identify duplicate references, i.e., those items that refer to the same reference. While these duplicate references may match exactly on certain metadata fields, misspellings or incorrect entries can introduce difficulties in identifying duplicates. This is most closely related to the field of *record linkage*, which was first explored in the 1950s and 1960s (Elmagarmid et al., 2007). One of the earliest and most straightforward approaches to record linkage is a rule-based approach where reasonable rules are manually developed to match the data in the different fields and subsequently refined when exceptions to the rules are found (Elmagarmid et al., 2007). Such approaches typically achieve high accuracy but at a cost of much manual effort and continuous updating of the rule base. Fellegi and Sunter (1969) were the first to pioneer probabilistic approaches to record linkage. Here, pre-labeled data sets are used to automatically determine the importance weights that should be assigned to the different metadata fields in the matching process. Different string similarity metrics are then used to determine the similarity between matching metadata fields. By setting thresholds on the combined matching score, pairs of records are then predicted to be duplicates or not (Fellegi and Sunter, 1969).

[4]This decision is supported by preliminary experiments using shingling and the SimHash algorithm to de-duplicate CiteULike references (Henzinger, 2006). These experiments showed inferior performance compared to state-of-the-art record linkage approaches.

In recent years, an increasing body of work focuses on the application of machine learning algorithms to automatically learn the field weights and similarity thresholds for accurate matching functions. State-of-the-art approaches include Parag and Domingos (2004) and Bilenko et al. (2003). The latter compared a variety of character-based and token-based string similarity metrics such as the Levenshtein distance and tf·idf weighting, and found that the choice of optimal metrics is largely dependent on the data set used. They also showed that learning accurate matching functions using Support Vector Machines can yield significant improvements over the traditional probabilistic approach. We follow the work of Bilenko et al. in our de-duplication approach and use machine learning to learn the accurate matching function for combining the different metadata field similarities.

The effect of duplicate content on recommender system performance has not been the topic of much study, but some studies have been carried out in the IR community. Bernstein and Zobel (2005) explored techniques similar to those used by Henzinger (2006) on the runs submitted to the TREC Novelty track and found that returning duplicate documents had a significant effect on search experience, with up to 17% of duplicates in the returned result lists. In Section 8.4 we perform the same kind of analysis for social bookmarking recommendation.

8.3 Duplicate Detection

As mentioned before, unless new posts contain a universal document identifier like a DOI number, no steps are taken by CiteULike to match these articles to existing items in the database. This can result in a large number of duplicates on CiteULike. A good example is the article "Collective Dynamics of Small-World Networks" (Watts and Strogatz, 1998). The original version has been added by 43 different users, but it has at least 28 duplicates, added by 29 different users. This means that over 40% of all users who added the article might actually be found by following links in the folksonomy from the original article to other users.

In this section we describe our two-step duplicate detection strategy for CiteULike. The first step in this strategy was obtaining training material to get an estimate of how widespread duplicates are on CiteULike, and to train our duplicate classifier on. We describe collection of this training material in Subsection 8.3.1. In Subsection 8.3.2 we describe the second step: how to construct the duplicate classifier. In Subsection 8.3.3 we present the experimental results of our classifier.

8.3.1 Creating a Training Set

The first step in analyzing the effect of duplicates on the recommendation process is constructing a gold standard training set to evaluate our de-duplication methods against. This training set should contain both examples of pairs of duplicate items on CiteULike and of pairs of unique, different items, because any multi-class discriminative machine learning algorithm presented with only positive examples can and will only learn to output a positive

prediction every time. Furthermore, any realistic collection of training material should also contain an appropriate ratio of duplicate pairs and different pairs, as social bookmarking systems in general are more likely to contain only a small percentage of duplicate content. Our collection process starts by (1) identifying likely pairs of duplicates and an appropriate number of pairs of distinct items, followed by (2) a manual annotation phase where all considered pairs were judged as being duplicate pairs or not.

We select our training set pairs as follows. First, we order all the items in our original CiteULike data set[5] on popularity counts and divide them into 10 logarithmic bins based on these frequency counts. From each bin we randomly pick 5 seed items—less if the bin contained less than 5 items—resulting in a total of 34 seed items. For each of these 'focus' items we extract all bigrams from their tokenized[6] titles that do not contain a stop word[7]. For examples, for an item with the title "Referral Web: Combining Social Networks and Collaborative Filtering" (Kautz et al., 1997) this results in the following 5 bigrams: `referral web`, `web combining`, `combining social`, `social networks`, and `collaborative filtering`. From each seed item's bigrams we randomly select two of them, which are then used as queries on all item titles in the CiteULike data set. This means that for each seed item two queries are constructed and matched against all 500,000+ titles in the CiteULike data set. Because duplicate items can often be the result of spelling errors, this matches a good number of duplicate items with different IDs that effectively represented the same article. In practice, these bigrams also match a wide variety of titles more often than not, resulting in a large number of pairs of distinct items. All items whose title matches one of these two bigram queries are then each paired up with the respective seed item(s) they originated from. This selection process found 2,683 matching items and a total of 2,777 pairs of seed items and matched items.

In the second phase of our collection process, we manually annotated these 2,777 pairs. Appendix D contains more information about the annotation process. Of the 2,777 annotated pairs, 189 were found to be duplicates of one of the 34 seed items, and 2,588 were not, giving us a training set with 6.8% duplicates. It is very likely that because of our bigram matching method—a straightforward form of de-duplication in itself—we have overestimated this proportion of duplicates in our training set. The actual ratio of positive to negative examples in the complete CiteULike data set is expected to be much lower. However, we do not believe that overestimating this proportion will hurt our detection method, as the use of sampling methods to artificially constrain the ratio of positive to negative examples is common in machine learning and is not dependent on the actual ratio. In addition, our bigram method will result in higher recall at the expense of precision. However, since we use human annotators to interactively check our training set for duplicates and non-duplicates afterwards, precision can be allowed to take a back seat to recall in our collection of the training set. Furthermore, the advantage of using the bigram queries is that the non-duplicate pairs are more likely to be more difficult to predict properly for our machine learning algorithm. If we were to select items randomly to pair up with the seed

[5]For this selection process we used the original, unfiltered CiteULike data set as described in Table 3.1 instead of the data set used in our experiments in Chapters 4–6 and described in Table 3.2.

[6]Regardless of the algorithm used, normalization of the data is very important and can result in higher accuracy.

[7]We filtered stop words using the SMART stop word list.

items, term overlap can be expected to be much lower than in our present case, which could overestimate the performance of the classifier.

For 8 of the 34 seed items we could not find any duplicate items. This does not mean that these could not have duplicates, merely that our bigram matching method did not find any. For the remaining 26 seed items the number of duplicates varies from 1 to 30, with 5 of of the 26 items having more than 25 duplicate versions on CiteULike. Figure 8.1 shows the distribution of duplicate counts over the seed items.

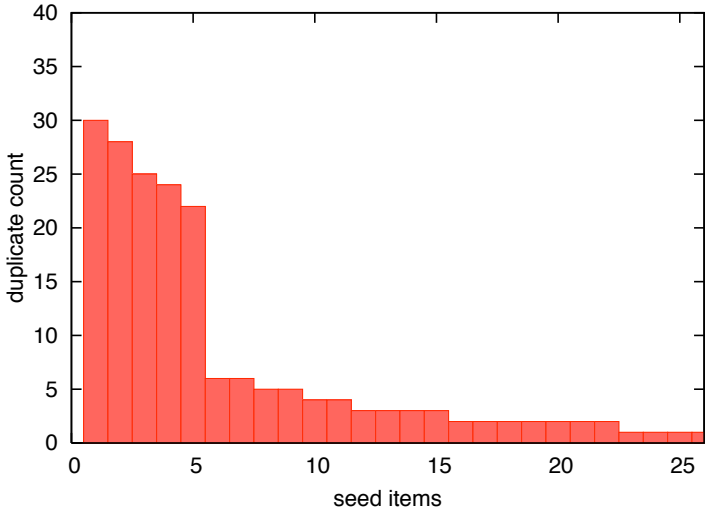


Figure 8.1: Distribution of the duplicates over the different seed items in our training set.

8.3.2 Constructing a Duplicate Item Classifier

The second step of our duplicate detection strategy for CiteULike involves constructing a duplicate item classifier. Our approach to duplicate detection is inspired by Bilenko et al. (2003) and involves a pairwise prediction task: we represent a pair of items by different features, where each feature represents some type of similarity measure calculated on the item pair. These instances are then fed to a machine learning algorithm that learns the best combination of feature weights, and predicts a binary label: duplicate pair or distinct pair of items.

Feature Representation What features do we use for duplicate classification? Scientific articles can be annotated with many different metadata fields in CiteULike, but the majority of these fields are only sparsely filled. Table 5.1 lists how much metadata we have available for each field. The following six fields are populated densely enough to be of use in matching all of the items in the CiteULike data set: `AUTHOR`, `ENTRYTYPE`, `TAGS`, `TITLE`, `URL`, `YEAR`. Inspection of the `ENTRYTYPE` metadata field, denoting the BibTeX entry type, showed it is not used consistently enough to be useful. As electronic versions of papers can be found in many different location on the Web, the `URL` field will not be reliable enough either. Finally, since tags represent a personal vocabulary, we do not use them as a source of metadata comparison either. This leaves us with the `AUTHOR`, `TITLE`, and `YEAR` fields to calculate the

similarity features. Bilenko et al. (2003) compared a variety of character-based and token-based string similarity metrics such as the Levenshtein distance and tf·idf weighting. We follow their approach to feature construction and calculate six different features for each feature pair: four for the TITLE field, and one each for the AUTHOR and YEAR fields. Table 8.1 describes these six features.

Table 8.1: Similarity features used in duplicate detection.

Feature	Field	Description
F_1	TITLE	Cosine similarity on length-normalized, unigram term vectors.
F_2	TITLE	Cosine similarity on length-normalized, bigram term vectors.
F_3	TITLE	Length of the longest common substring divided by the length of the longest string.
F_4	TITLE	Normalized Levenshtein distance on the titles.
F_5	AUTHOR	Unigram term overlap of tokenized author string.
F_6	YEAR	Binary match on publication year.

All feature values are mapped to the $[0,1]$ range. Note that the first three title-based features are arranged in order of increasing strictness. Feature F_1 calculates the cosine similarity on unigram term vectors of the two item titles, while feature F_2 uses bigram term vectors. A high score on F_2 automatically means a high score on F_1. A high value for feature F_3 implies high values for F_1 and F_2, and additionally requires that the unigrams and bigrams occurring in the longest common substring are also in the right order. The normalized Levenshtein distance can be calculated in different ways; we calculate it according to Navarro (2001) as $1 - \frac{Levenshtein(title_1,title_2)}{max\_length(title_1,title_2)}$. From our experiences with the training set, sometimes a mistake separating two duplicates can be as simple as entering the year 2003 instead of 2004. Generally speaking, however, relaxing the binary matching on years is likely to result in too many false positives. Figure 8.2 shows some examples of training set instances represented using our six features.

ITEM ID 1	ITEM ID 2	FEATURE 1	FEATURE 2	FEATURE 3	FEATURE 4	FEATURE 5	FEATURE 6	CLASS LABEL
99	1075248	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1
99	1075282	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1
99	1091096	0.2981	0.0000	0.1111	0.4677	0.0000	1.0000	0
99999	1791032	0.3333	0.2582	0.1250	0.1695	0.0000	1.0000	0
1272533	1412129	0.2390	0.1508	0.1667	0.2432	0.0000	1.0000	0
1272533	1416466	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1
1272533	1821589	1.0000	1.0000	1.0000	1.0000	0.0000	1.0000	1

PAIRED ITEMS — DUPLICATES?

Figure 8.2: Example instances from our duplicate detection training set. The first instance shows a perfect match on all features, which is reflected in the '1' value for the class label, whereas the fourth instance matches on publication year, but not well on the other features. The seventh instance is an example of two paired items that represent the same content, but where the author field was not filled in for one of the users.

Experimental Setup The next step in constructing a duplicate item classifier is training it on the training set we constructed in the previous section. For each of the 2,777 pairs in the training set, we create an instance with the six similarity features calculated on the item pair and the class label '1' or '0', denoting a duplicate or distinct pair of items. When we train a machine learning algorithm on this data set, it attempts to learn the optimal set of combination weights to detect duplicate pairs. Like Bilenko et al. (2003), we use Support Vector Machines (SVMs) as our machine learning algorithm. SVM is a supervised machine learning algorithm that attempt to find the maximum margin hyperplane separating the two classes of pairs. We refer to Burges (1998) for more detailed information about the algorithm. We used the publicly available SVM^{light} implementation for our experiments[8]. Similar to our experimental setup for our recommendation experiments, we divided our 2,777 pair data set into a training and a test set, and randomly assigned 15% of the pairs to a test set and the remaining 85% to a training set. We further subdivided our training set into 10 folds for 10-fold cross-validation. We optimized the parameters of SVM^{light} using our 10-fold cross-validation setup with *Paramsearch*, which is a publicly available implementation[9] of the wrapped progressive sampling approach to parameter optimization proposed by Van den Bosch (2004). *Paramsearch* optimizes the four SVM^{light} parameters listed in Table 8.2.

Table 8.2: List of parameters optimized for the SVM classifier by *Paramsearch*.

Parameter	Description
`-t`	Type of kernel function (linear, polynomial, radial basis function, or sigmoid).
`-g`	Gamma parameter for the radial basis kernel.
`-t`	Trade-off between training error and margin.
`-j`	Cost-factor by which training errors on positive examples outweigh errors on negative examples.

In addition to these parameters, we also experiment with different ratios of positive and negative examples. Data sets that are heavily skewed towards a single class can be problematic to learn for a machine learning algorithm when evaluated on, for instance, accuracy. The algorithm might simply learn a classifier that outputs the majority class label all the time, if that results in a high accuracy score. Even when evaluating using a proper evaluation metric such AUC instead of accuracy, a heavily skewed data set still might not result in the best possible classifier. Another solution to this problem is adjusting the ratio of positive to negative instances in the training set. By downsampling the majority class, we can 'force' the classifier to base its decision purely on the feature value and not just on the evaluation metric (Japkowicz and Stephen, 2002). This has been shown to work well on other tasks and data sets[10]. We therefore construct two alternative data sets by randomly sampling the positive and negative examples in a 1:1 ratio and a 1:2 ratio, where the data sets contain one respectively two negative instances—i.e., distinct pairs—for each positive one—i.e., a duplicate pair. We refer to these data sets as PosNeg 1-1 and PosNeg 1-2. Such instance sampling reduces the size of the training set. To show that the reduction does not simply make the classification problem easier, we also constructed a data set of the same size as

[8] Available at `http://svmlight.joachims.org/`

[9] Available at `http://ilk.uvt.nl/paramsearch/`

[10] Certain machine learning algorithms are more susceptible to this than others, but SVMs have also been shown to be affected by this problem (Japkowicz and Stephen, 2002).

POSNEG 1-1 but with randomly selected instance from the entire data set ALLINSTANCES. Performance on this new data set, SIZE 1-1, will show if the downsampling simply made the problem easier by reducing the size, or if the sampling is a successful technique. If the scores on POSNEG 1-1 are significantly higher than on SIZE 1-1, then the downsampling improved performance because of a more optimal instance ratio, and not because of the reduced data set size. We do the same for the POSNEG 1-2 and create a matching SIZE 1-2 data set. For each of these extra four data sets we also partitioned the data as described above and ran *Paramsearch* to find the optimal parameters settings. The optimal settings were then used to train the model on all of the training data and used to classify the test data. The results of these experiments are listed in the next section.

Duplicate Identification in CiteULike After training a duplicate classifier we can use it to detect duplicates in our entire CiteULike data set. We use the unfiltered data set containing 585,351 items, so that we can gather complete frequency information about all items, and not just those that occur at least twice. Ideally, we would want to determine for each pair of items in our item set L_{ALL} whether or not they are duplicates. To do this we represent each item pair in the same format as we did with our training set, as described in Table 8.1 and Figure 8.2, but without the class label which we are trying to predict. Unfortunately, with a data set size $|L_{ALL}|$ of 585,351 items considering all possible pairwise comparisons requires examining $\frac{1}{2}L_{ALL}(L_{ALL}-1)$ item pairs, which amounts to 171,317,603,925 comparisons.

This is clearly an undesirable situation, but also a common one in the field of record linkage. While the total number of comparisons might be impractical to carry out, most record pairs are clearly non-duplicates, so they do not need to be considered for matching. This can be achieved through *blocking* by dividing the database records in blocks, where only the records within the same block are compared to each other (Bilenko et al., 2003). An example of a blocking method is the *sorted neighborhood method*, which sorts the records over the most important discriminating attribute of the data, such as the TITLE field in our case. After the sort, the comparison of records is then restricted to a small neighborhood within the sorted list (Hernández and Stolfo, 1995). However, we believe that the TITLE field is also the most vulnerable to spelling errors, so we suspect sorted neighborhood blocking would result in too many missed duplicate pairs. We therefore apply our own blocking method: *popularity blocking*, based on the popularity counts of the items. We identify all duplicates in three different steps, as visualized in Figure 8.3.

We can reasonably assume that for an item with duplicates, the original item will often be the most popular compared to its clones. We therefore divide our items into two popularity groups: the set of popular or *high range* L_{HIGH} items that have been added to 10 or more user profiles, and the less popular, *low range* set of items L_{LOW} that contains all the remaining items. This results in 680 and 584,671 items in the high and low range sets respectively. It is reasonable to assume that the original item of a set of duplicates usually falls into the high range, while most of its duplicates fall into the low popularity range. In the first step of our deduplication process we do a pairwise comparison of each item in L_{HIGH} with each item in L_{LOW}, resulting in 397,576.280 comparisons. Then, in the second step, we compare all the high popularity items with each other, resulting in 230,860 comparisons for our data set. We compare the high range items with each other on the off-chance that one of the duplicates is very popular as well. Finally, we select all of the duplicate items found in the first two

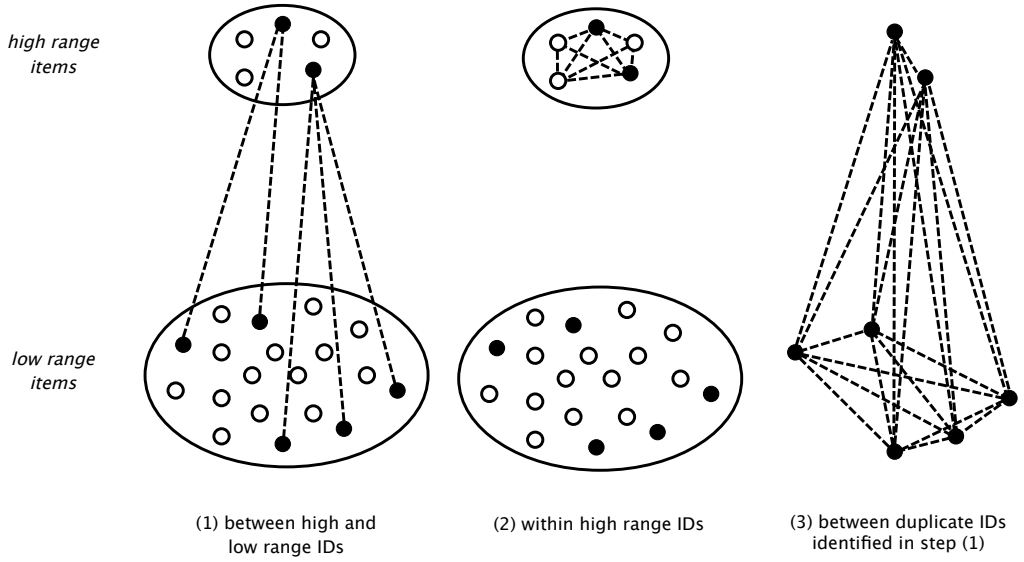


Figure 8.3: Visualization of our popularity blocking scheme for duplicate identification. Items, represented by the small circles, are divided into two groups based on popularity. Solid black circles represent items for which a duplicate has been identified, with the dotted line connecting a pair of duplicate items.

steps and classify all pairs within this group. We take this third step to detect the duplicates we missed by not comparing the low popularity items with each other, resulting in another 19,279,888 comparisons. In total, we performed 417,087,028 comparisons, which is 99.7% of the total number of possible comparisons.

8.3.3 Results and Analysis

Evaluating the Duplicate Item Classifier Table 8.3 lists the results of our different duplicate classification experiments on the training set. What we see is that the PosNeg 1-1 data set, where positive and negative instances were sampled in an equal ratio, achieves the best classification performance, at an F-score of 0.9804 and an AUC score of 0.9795. For F-score, the PosNeg 1-1 data set is significantly better than all other variations at $\alpha = 0.05$. The PosNeg 1-2 variation, however, does not perform significantly better than the using all instances in the training set. Both PosNeg variants perform significantly better as measured by F-score than their corresponding Size runs, which shows that the improved performance is not just due to a reduced size of the training set. When we measure classification performance by AUC, none of the differences are statistically significant but we still see the same pattern occurring as we do for F-score.

Classifying Duplicates on CiteULike Based on the results in Table 8.3, we select the PosNeg 1-1 classifier for classifying the duplicate items in the entire CiteULike data set. The optimized parameter settings for this SVM-based classifier include using a radial basis function kernel (`-t`) with the γ parameter (`-g`) set to 0.128. The cost factor (`-j`) is set to the default value of 1.0, and the training error-margin trade-off (`-c`) is set to 5. Table 8.4 shows the number of duplicate pairs identified in each of the three steps shown in Figure 8.3.

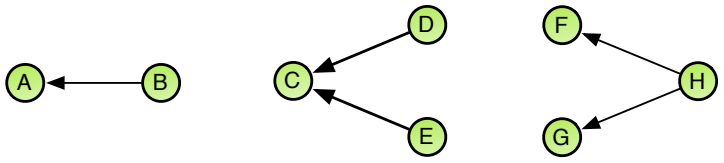
Table 8.3: Results of duplicate classification on the training set, evaluated on F-score and AUC. Best scores are printed in bold.

Data set	F-score	AUC
AllInstances	0.9274	0.9532
PosNeg 1-1	**0.9804**	**0.9795**
PosNeg 1-2	0.9393	0.9531
Size 1-1	0.5537	0.9164
Size 1-2	0.5359	0.9195

Table 8.4: Results of deduplicating the CiteULike data set.

Step	Duplicate pairs	Total pairs	% of pairs classified as duplicates
Step 1	30,421	397,576.280	0.0077%
Step 2	752	230,860	0.3257%
Step 3	172,814	19,279,888	0.8963%
Total	19,374	417,087,028	0.0046%

Certain pairs examined in step (3) were already compared in steps (1) and (2); filtering out these identical pairs left us with a total of 19,374 pairs of duplicate items, which is why the total duplicate count is lower than the sum of the duplicates uncovered in the three individual steps. Overall, of the 400+ million pairs we compared, a little under 0.005% were classified as duplicates. This shows that duplicate identification is like trying to find a needle in a haystack. The largest proportion of duplicates were detected in step 3, which is to be expected: this set already contains the duplicates detected in the first two steps, and additionally links unpopular duplicates of the same original, popular items. The final duplicate percentage is several orders of magnitude smaller than the the percentage in our training set. While we already reported that number to be an overestimate, the actual percentage of duplicates is still lower than we expected. Figure 8.4 shows three examples of basic graph patterns of how these 19,374 duplicate pairs tend to be connected to each other.



(a) (b) (c)

Figure 8.4: Example duplicate graph patterns

Pattern 8.4(a) is the standard pattern, where item B was classified as a duplicate of item A. For each pair we wish to replace the newest item by the oldest item, i.e., we consider the item that was added first to CiteULike to be the original item. The arrow pointing from item B to item A denotes that item A is the oldest known version of the item, and that all occurrences of item B should therefore be replaced by item A. Pattern 8.4(b) occurs when

multiple items are all classified as duplicates of the same item. In this case, items D and E are both duplicates of original item C, and should both be replaced by item C in the data set. Both patterns do not present any problems and are straightforward to apply. Pattern 8.4(c), however, represents a possible dilemma: item H is identified as the duplicate of two different items. One option is to use the pair with the oldest original item, i.e., replace item H by item F. It is more likely, however, that our classifier failed to identify a link between items F and G, because not all possible item pairs were examined. In this case, we replace this pattern by links to the oldest item F from all the other items in the pattern—items G and H. After this re-alignment, items G and H are both replaced by item F. We only performed this alignment step on this third pattern type once for each pattern. Errors made by our duplicate classifier could result in introducing links between items that are not duplicates and could results in chaining together too many original, yet different items if we decided to iterate this alignment step until no more changes are made. After this step, 9,557 duplicate pairs were left for deduplicating our data set.

Let us take a closer look at these duplicates we uncovered. Figure 8.5(a) plots the popularity of the duplicates that were identified in the CiteULike data set, and shows it resembles a Zipfian distribution. It shows that the mistake of creating a duplicate article on CiteULike is often propagated , as many duplicates are added multiple times. One specific duplicates was added over 80 times.

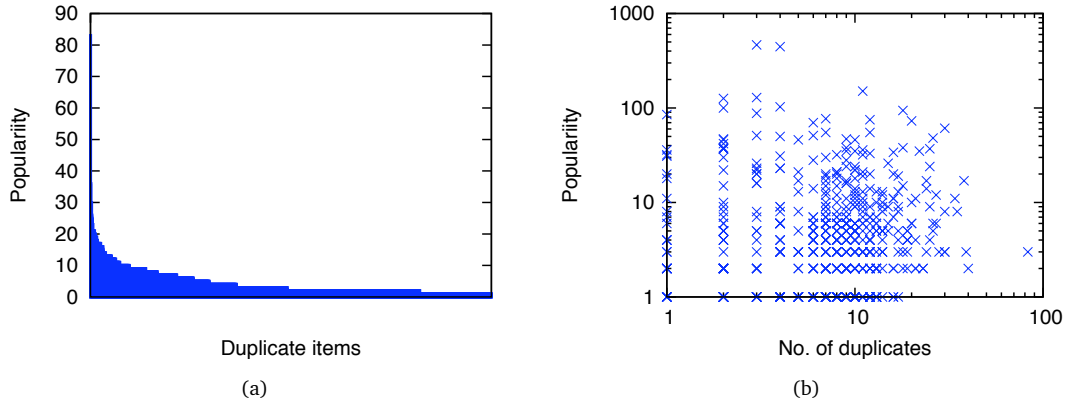


Figure 8.5: Visualizing the identified duplicates. Figure 8.5(a) plots the popularity of the duplicates that were identified in the CiteULike data set, whereas Figure 8.5(b) plots the popularity of all original, first versions of an item against the number of duplicates that were detected for them.

Figure 8.5(b) plots the popularity of all original, first versions of an item against the number of duplicates that were detected for these originals, and shows no apparent relation between these two properties. It shows that, contrary to what we expected in Section 8.1, popular items do not automatically beget more duplicates, as the number of duplicates for a given article is not commensurate with the popularity of the first added version of that article. Perhaps we were wrong in assuming that there is a fixed probability that spelling errors or missing DOIs cause a duplicate article to be spawned.

8.4 The Influence of Duplicates on Recommendation

In the previous section, we described an approach to detecting duplicate items in a social bookmarking system. Just like spam, duplicate content can be a nuisance to users, duplicate items can pollute the search results and prevent users from reaching all related content when browsing the folksonomy. In addition, as with spam content, storing duplicate items also wastes computing power and storage capacity that could be put to better use. Detecting and removing duplicate content is therefore another vital component of a mature social bookmarking service. We believe proper deduplication can be equally beneficial to recommendation. User satisfaction is not served by multiple versions of the same item showing up in the lists of recommended items, and a leaner version of the bookmarks database will also cut down on computation time. The aim of this section is analogous to that of Section 7.4 in the previous chapter: in a small case study on one of our data sets we wish to examine how much recommendation effectiveness is weakened by the presence of duplicate content. We know of no other related work on this problem, and an initial investigation of this issue will therefore be our contribution here.

8.4.1 Experimental Setup

We test the influence of duplicate content on recommendation using our CiteULike collection, and we described in the previous section how we detected the pairs of duplicate items in this collection. After the duplicates have been identified they need to be filtered out, and this can be done at two different stages: (1) on the original input data set (*eager deduplication*), and (2) by removing the duplicate items from list of generated recommendations (*lazy deduplication*). We deduplicate the original data set by replacing all newer items of each duplicate pair by their original item until there are no more items left to replace. This typically takes 5 to 7 iterations.

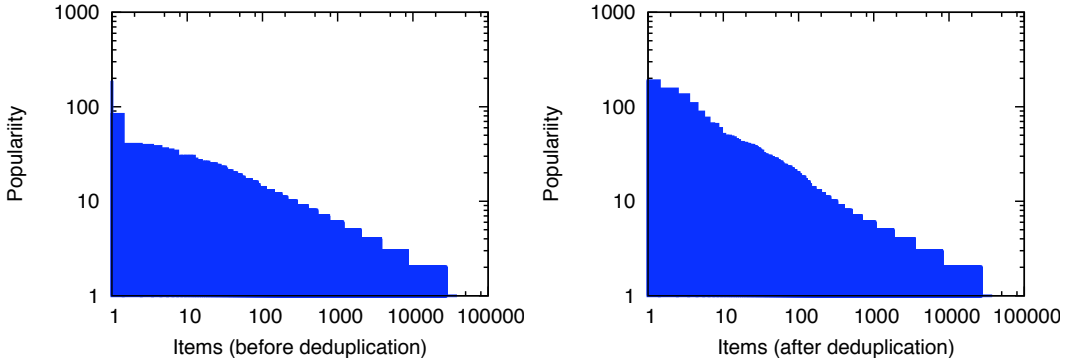


Figure 8.6: Popularity of the CiteULike items before (left) and after deduplication (right).

Figure 8.6 shows the popularity of all items in the CiteULike data set before and after deduplication. Plotting the two quantities on a log-log scale shows that both plots resemble a Zipfian distribution, with the curve before deduplication being slightly flatter at the top. Deduplication apparently removes this as the graph on the right shows a stronger Zipfian distribution.

To make for a fair comparison, we used our filtered version of the CiteULike data set as described in Subsection 3.4.1, and applied our deduplication procedure on this data set. This means we stay as close as possible to retaining users with 20 items or more, and items occurring in at least 2 user profiles. Using a data set that does not filter out the unique items could see a greater reduction in items when deduplicating, but such a filtering threshold would lead to other problems, such as those discussed in Subsection 3.4.1. In addition, we also deduplicated the relevance judgments for this deduplicated data set. Table 8.5 offers a basic description of the new, deduplicated CiteULike data set compared to the original one.

Table 8.5: Basic statistics of our deduplicated CiteULike data set compared to the original version.

	Original data set	Deduplicated data set
# users	1,322	1,322
# items	38,419	37,339
# tags	28,312	28,196
# posts	84,637	83,439
avg # items per user	64.0	63.1
avg # users per item	2.2	2.2
avg # tags per user	57.3	57.0
avg # users per tag	2.7	2.7
avg # tags per item	5.3	5.4
avg # items per tag	7.3	7.1
sparsity	99.8334	99.9236

If we had a perfect deduplication algorithm, we would see a decrease in item count as duplicate items are rolled back into their earliest, original entries. The number of posts by a user would stay the same here; only the item identifier would be altered. We can assume that very few users would make the mistake of adding duplicate versions of the same resource to their own profile, so the total number of posts in the data set would stay almost the same[11]. Overall sparsity of the data set would decrease however. As we see in Table 8.5, this is not the case. This indicates that our deduplication algorithm is too aggressive in merging duplicate pairs: too many pairs of items within user profiles are misclassified as duplicates and replaced by a single post. We will discuss the ramifications of this later on in this section, as well as possible solutions.

Although it would be preferable to deduplicate the entire data set to reduce storage and processing requirements, it is also possible to perform deduplication at another stage in the recommendation process. Similar to what Bernstein and Zobel (2005) do for document search, we can also leave the duplicates in the data set and instead deduplicate the output list of recommendations. This is the second deduplication method we investigate.

Which algorithms do we use to test to determine the influence of duplicates? As in the previous chapter, we select the best-performing approaches from chapters 4 and 5. The best-performing folksonomic recommendation algorithm was the standard item-based CF algorithm with the I-BIN-IDF-SIM similarity metric, i.e., usage-based overlap using *idf*-weighting.

[11] This phenomenon does occur, however, and the posting count would therefore decrease slightly even if we used a perfect deduplication algorithm.

We optimize the neighborhood size for the spam extended data sets using our 10-fold cross-validation setup. We select profile-centric content-based filtering as the best metadata-approach and use all intrinsic and extrinsic metadata fields for our user and item representations. Using the deduplicated version of our data set, we consider all users and items for calculating the similarities. We evaluate on the same users with the same test items as we did in our previous recommendation experiments on the CiteULike data set.

What do we expect from these two types of algorithms? We expect a metadata-based approach to suffer from the presence of duplicates, since retrieving any item based on the user's profile would also mean retrieving all of the duplicates with near-identical metadata. The presence of duplicates could also be detrimental to performance of an item-based CF approach. If there is an item with duplicates that are relatively popular themselves, these duplicates and the original item are likely to be divided over the users interested in the original item. This means that there is no guarantee that all the best nearest neighboring items are located for each item, resulting in suboptimal performance. However, duplicate versions of the same item are not likely to show up in the same list of recommendations, because usually they are not as popular as the original versions. This means there is not enough evidence to recommend them over other items. We expect that a metadata-based approach is therefore more vulnerable to spam.

8.4.2 Results and Analysis

Table 8.6 shows the results of our two selected algorithms on the CiteULike data set with deduplication on the original data set and deduplication of the list of recommendations. We find only small changes in performance of our two deduplication methods compared to the MAP scores on the original CiteULike data set. For the item-based CF approach, the MAP score drops slightly from 0.0887 to 0.0882 and 0.0886; for the profile-centric approach, deduplicating the entire data set causes a slight drop from 0.0987 to 0.0985, whereas deduplicating the result list shows a small increase in MAP score to 0.1011. None of the changes are statistically significant.

Table 8.6: The influence of duplicate items on recommendation performance on our original CiteULike data set. Deduplication was at two different stages: on the input data set and on the result list. Reported are MAP scores and best results are printed in bold.

	Item-based CF	**Profile-centric**
Original CiteULike data set	**0.0887**	0.0987
Deduplication on data set	0.0882	0.0985
Deduplication on result list	0.0886	**0.1011**

Figure 8.7 shows the same picture in the per-user differences in Average Precision (AP) for the four different deduplication runs. For all but the profile-centric method with deduplication of the result lists, we find that deduplication provides benefits for some users, but that misclassifications introduce errors for others. For the profile-centric method with deduplication of the result lists we see a slightly more positive effect, although the differences are too small to be significant or useful.

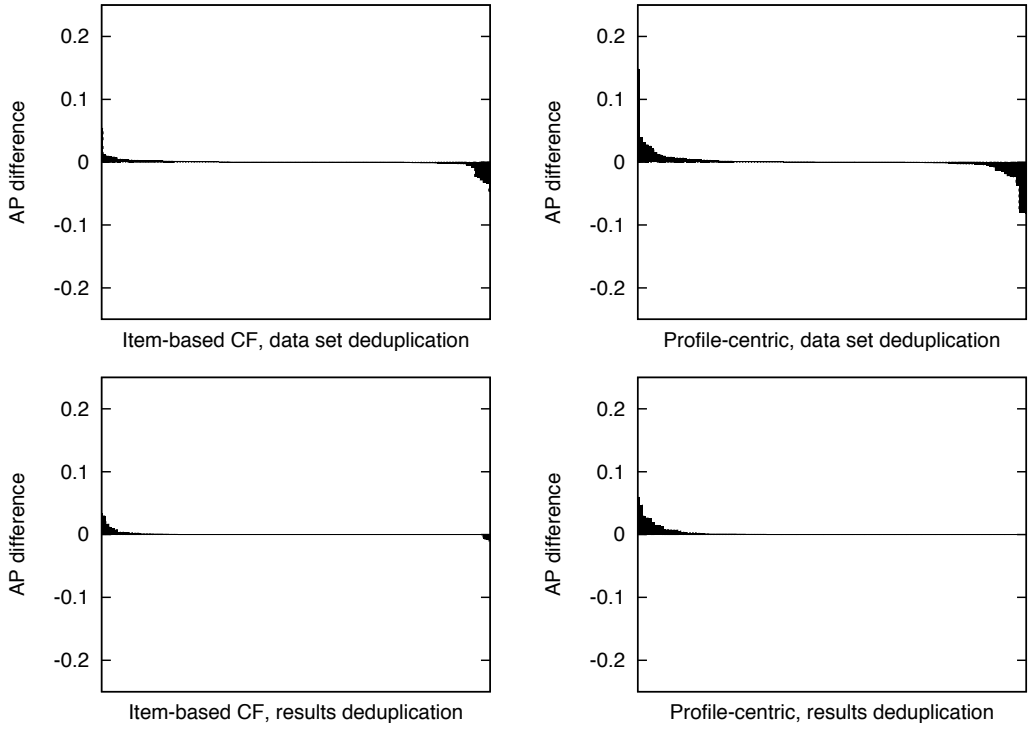


Figure 8.7: The top level figures show the performance of our two algorithms when the CiteULike data set is deduplicated; the bottom two figures show the results of deduplication of the recommendation lists. The figures on the left correspond to the item-based CF algorithm and the figures on the right correspond to the profile-centric recommendation algorithm.

We expected the metadata-based algorithm to show the greatest benefits from removing all duplicate content but we do not find any strong evidence for this. One reason for this is that we find less duplicates than we intuitively expected, which automatically lowers the possible effect deduplication could have. In addition, we believe that too many of the duplicate pairs we did identify within user profiles are misclassified as duplicates and replaced by a single item. While our classifier performs well on a 1:1 downsampled data set, the actual percentage of duplicates on CiteULike is orders of magnitude lower. We believe our downsampling might have been too much in favor of the positive instances, which caused the classifier to generate too many false positives on the complete CiteULike data set. There are several things that could be improved upon in future experiments:

- One of the most straightforward ways of increasing the performance of the duplicate classifier is increasing the size of our training set. In the past prediction accuracy has been shown to keep increasing with the size of the training set (Brill and Banko, 2001). Adding more training examples means the classifier will get better at recognizing the different ways that duplicates 'sneak' into the system. It will also result in a a data set with a more realistic proportion of duplicates. Perhaps with such a data set, another ratio of positive to negative examples would also provide better performance than the 1:1 ratio we found in Subsection 8.3.3.

- A cursory glance at some of the misclassified pairs reveals cases where a high title similarity was caused by an overlap in phrases such as "The history of the", but then with differing final nouns. Future work should look into weighting different types of words, such as determiners and prepositions, when matching phrases, or possibly filtering out the stop words altogether. Some mistakes are hard to avoid however: for instance, we found two genuinely different items in our data set that were both titled "The Semantic Web", but written by different authors in different years. With a strong focus on features that represent the `TITLE` field, however, these will almost always be classified as false positives.
- We used a blocking scheme based on item popularity to limit the number of comparisons, but perhaps a sorted neighborhood scheme would result in less false negatives and improve the quality of the classifier.

While we are not able to provide a conclusive, positive answer about the influence of deduplication on recommendation, we believe the task itself is still a vital one. Future work that takes the above suggestions into consideration should be able to give a more precise picture of the influence of duplicate content on social bookmarking websites.

8.5 Chapter Conclusions and Answer to RQ 5

In this chapter we attempted to determine what the influence of duplicate content is on social bookmarking systems, as covered by our fifth and final research question and its two subquestions.

RQ 5 How big of a problem is the entry of duplicate content for social bookmarking services?

RQ 5a Can we construct an algorithm for automatic duplicate detection?

RQ 5b What influence do duplicates have on recommendation performance?

We argued that duplicate entries of the same item can cause the same kind of problems for social bookmarking as spam content. To quantify the problem we examined our data set collected from the CiteULike website, which does not do extensive deduplication of data entered by the users. We collected a small training set of nearly 3,000 item pairs and annotated them manually as being duplicates or not. This training set was then used to construct a duplicate item classifier for this pairwise detection task with acceptable performance at first glance (RQ 5a). Because of the large size of our complete CiteULike data set, we employed a heuristic method called popularity blocking to bring the number of necessary classifications down to manageable proportions. Classifying the entire CiteULike data set then resulted in the detection of over 9,000 duplicate pairs. We found no relation between the popularity of an original item and the number of duplicates it spawns (RQ 5). Finally, we took a closer look at the influence of duplicate on recommendation performance by examining the effects

of deduplication on the data set and deduplication of the list of resulting recommendations (RQ 5b). We tested a CF and a content-based approach and found that deduplication was not beneficial for either algorithm or deduplication option. We believe our duplicate item classifier was not good enough at identifying the real duplicate pairs and that this is the reason for our negative results. A larger training set, a better training regimen (e.g., an optimized downsampling ratio), and better similarity features should be able to remedy this problem in future work.

Conclusion

In Part I of the thesis, we covered the core of our recommendation experiments, and examined how to incorporate information from the folksonomy and metadata into recommendation algorithms. In addition, we examined if it was possible to combine the recommendations generated by our algorithms and improve recommendation performance by exploiting the complementarity of our different algorithms. In Part II, we zoomed in on two specific growing pains that social bookmarking services encounter as they become more popular: spam and duplicate content. We quantified how common these problems are, and proposed algorithms for the automatic detection of spam and duplicate content. We also investigated the influence they can have on recommending items for social bookmarking websites.

We come to a conclusion of this thesis in this final part. We recall our five research questions and summarize the answers to each of them. We provide practical guidelines for implementing a recommender system on a social bookmarking website. Then we list what we believe to be our main contributions with this work, and we conclude with recommendations for future research.

Chapter 9

Discussion and Conclusions

In this thesis we have examined how recommender systems can be applied to the domain of social bookmarking. More specifically, we have investigated the task of item recommendation, where interesting and relevant items—bookmarks or scientific articles—are retrieved and recommended to the user, based on a variety of information sources about and characteristics of the user and the items. The recommendation algorithms we proposed were based on two different characteristics: the usage data contained in the folksonomy, and the metadata describing the bookmarks or articles on a social bookmarking website. We formulated the following problem statement for the thesis in Chapter 1.

PS *How can the characteristics of social bookmarking websites be exploited to produce the best possible item recommendations for users?*

In this chapter we conclude the thesis. We start by answering our five main research questions in Section 9.1. In Section 9.2 we offer a set of practical recommendations for social bookmarking services seeking to implement a recommender system. We summarize our five main contributions in Section 9.3. We conclude this chapter in Section 9.4, where we list future research directions, drawing on the work described in this thesis.

9.1 Answers to Research Questions

Our problem statement led us to formulate five main research questions. Along the way, seven additional subquestions were formulated as well. In this section we summarize the answers to those twelve research questions. The first three research questions focused on how the two key characteristics of social bookmarking websites—the folksonomy and the metadata—can be utilized to produce the best possible recommendations for users?

RQ 1 How can we use the information represented by the folksonomy to support and improve recommendation performance?

In answering our first research question we focused on using the tags present in the broad folksonomy of social bookmarking systems, which describe the content of an item and can therefore be used in determining the similarity between two objects. We extended a standard nearest-neighbor CF algorithm with different tag similarity metrics to produce our TOBCF and TIBCF algorithms. We found that the performance of item-based filtering can be improved by using the item similarities based on the overlap in the tags assigned to those items (TOBCF). The reason for this is reduced sparsity in the item profile vectors; something we did not find in user-based TOBCF, which as a result did not benefit from using tag similarity. Bookmark recommendation is affected more strongly than reference recommendation. Our TIBCF algorithms did not produce competitive results. We may therefore conclude that tagging intensity is not a good measure of user and item similarity.

An examination of merging different types of similarity in our SimFuseCF algorithm yielded inconclusive results. We compared our algorithms with two state-of-the-art GBCF approaches: a graph-based algorithm using random walks and a tag-aware fusion algorithm. Here, we observed that our algorithm outperformed the random walk algorithm. The tag-aware fusion approach outperformed our own TBCF algorithms by fusing different representations and algorithms. On the basis of these results we may conclude (1) that tags can be used successfully to improve performance, and (2) that usage data and tagging data have to be combined to achieve the best performance.

RQ 2 How can we use the item metadata available in social bookmarking systems to provide accurate recommendations to users?

To answer RQ 2, we proposed four different algorithms, divided into two classes: two content-based filtering approaches and two hybrid approaches. In content-based filtering, a profile-centric approach, where all of the metadata assigned by a user is matched against metadata representations of the items, performed better than matching posts with each other because of sparseness issues. We also compared two hybrid CF approaches that used the metadata representations to calculate the user and item similarities. Here, we found that item-based filtering with the metadata-derived similarities performed best. What the best overall metadata-based algorithm is, is dependent on the data set. In Chapter 5, we formulated the following two additional subquestions.

RQ 2a What type of metadata works best for item recommendation?

RQ 2b How does content-based filtering using metadata compare with folksonomic recommendations?

We found that while sparsity of the metadata field does have an influence on recommendation performance, the quality of the information is just as important. Based on the experimental results we may conclude that combining all intrinsic metadata fields together tends to give the best performance, whereas extrinsic information, i.e., information not directly related to the content, does not (RQ 2a). Finally, compared to the folksonomic recommendation algorithms proposed, recommending using metadata works better on three of our

four data sets. Hence, we may conclude that it is viable choice for recommendation despite being underrepresented in the related work so far (RQ 2b).

RQ 3 Can we improve performance by combining the recommendations generated by different algorithms?

The answer to this third research question is positive: combining different recommendation runs yielded better performance compared to the individual runs on all data sets (RQ 3). We formulated an additional, more specific research question in Chapter 6.

RQ 3a What is the best recipe for combining the different recommendation algorithms?

In our experiments, weighted fusion methods consistently outperformed the unweighted ones, because not every run contributes equally to the final result. A second ingredient for successful fusion is using a combination method that rewards documents that show up in more of the individual runs, harnessing the *Chorus/Authority* effect, and improving the ranking of those items retrieved by more runs. Thirdly, the most successful combinations came from fusing the results of recommendation algorithms and representations that touched upon different aspects of the item recommendation process. Hence we may conclude that the theory underlying data fusion in IR is confirmed for recommender systems.

Our first three research questions were answered by taking a quantitative, system-based approach to evaluation, i.e., we simulated the user's interaction with our proposed recommendation algorithms in a laboratory setting. Such an idealized perspective does not take into account the growing pains that accompany the increasing popularity of social bookmarking websites: spam and duplicate content. We focused on these problems by formulating RQ 4 and RQ 5. The fourth research question addresses the problem of spam.

RQ 4 How big a problem is spam for social bookmarking services?

We examined two of our collections, CiteULike and BibSonomy, for spam and found that these data sets contain large amounts of spam, ranging from 30% to 93% of all users marked as spammers (RQ 4). We formulated two additional research questions in Chapter 7.

RQ 4a Can we automatically detect spam content?

RQ 4b What influence does spam have on the recommendation performance?

We showed that it is possible to train a classifier to detect spam users in a social bookmarking system automatically by comparing all of the metadata they have added together, to the metadata added by genuine users and by other spammers. This is best done at the user level of granularity instead of at a more fine-grained level (RQ 4a). Finally, we examined the influence of spam on recommendation performance by extending our BibSonomy bookmarks

data set with the spam entries we filtered out earlier. We tested a collaborative filtering and a content-based approach and found that spam has a negative effect on recommendation. Based on our experimental results we may conclude that the content-based approach was affected most by the spam presence. However, all result lists were unacceptably polluted with spam items, proving the necessity of adequate spam detection techniques (RQ 4b).

To address the problem of duplicate content on social bookmarking websites, we formulated the fifth research question and two additional, more specific research questions in Chapter 8.

RQ 5 How big a problem is the entry of duplicate content for social bookmarking services?

RQ 5a Can we construct an algorithm for automatic duplicate detection?

RQ 5b What influence do duplicates have on recommendation performance?

We examined one of our data sets, CiteULike, to quantify the problem of duplicate content (RQ 5). We constructed a training set and trained a duplicate identification classifier which found a small percentage of duplicates (RQ 5a). We found that these duplicate items follow a Zipfian distribution with a long tail, just as regular items do, which means that certain duplicates can be quite widespread (RQ 5). Finally, we examined the influence of duplicates on recommendation performance by creating a deduplicated version of our CiteULike data set. We tested a collaborative filtering and a content-based approach, but did not find any clear effect of deduplication on recommendation, because our duplicate identification classifier was not sufficiently adequate (RQ 5b).

9.2 Recommendations for Recommendation

Based on the answers to our research questions, we can offer a set of recommendations for social bookmarking services seeking to implement a recommender system. Note that our findings are specific to the task of recommending relevant items to users based on their profile; we cannot guarantee that our recommendations hold for other tasks such as personalized search or filling out reference lists.

Social bookmarking websites have two options at their disposal that both work equally well: recommending items using only the broad folksonomy or using the metadata assigned to the items to produce recommendations. The latter option of recommendation based on metadata is a good option when the website already has a good search infrastructure in place. In that case it is relatively easy to implement a metadata-based recommender system. An added advantage of having a good search engine is that it is also useful for detecting spam users and content with high accuracy. Recommendation using the information contained in the folksonomy is a good approach as well: here, we recommend implementing an item-based filtering algorithm that uses tag overlap between items to calculate the similarity.

However, since not all users tag their items, it would be even better to merge the tag information with the usage information before calculating the item similarities as suggested by Tso-Sutter et al. (2008). Depending on efficient implementation, performance can be greatly increased by combining the recommendations of different algorithms before they are presented to the user. It is important here to combine approaches that focus on different aspects of the task, such as different representations or different algorithms, preferably all.

To provide the user with a satisfactory experience it is important to perform spam and duplicate detection. While they may not influence the recommendations to a strong degree, their presence in the results list can be enough to for a user to lose trust in the recommender system. This illustrates that the success of any recommender system depends on the users, and whether or not *they* are satisfied with the system as a whole. Proper user testing of the system is therefore essential, and we will come back to this in Section 9.4.

9.3 Summary of Contributions

In this thesis we performed a principled investigation of the usefulness of different algorithms and information sources for recommending relevant items to users of social bookmarking services. Below we list the following five contributions we have made.

1. We examined different ways of using the information present in a folksonomy for recommendation, by extending a standard class of Collaborative Filtering algorithms with information from the folksonomy. These extensions were then compared to other state-of-the-art approaches, and shown to be competitive.

2. We determined the best way of using item metadata for recommendation, and proposed several new and hybrid algorithms. These algorithms were shown to be competitive with the more popular usage-based approaches that use the folksonomy. We were the first to perform such a comparison of content-based recommendation with collaborative filtering for social bookmarking services.

3. Compared to related work, we took a critical look at different methods for combining recommendations from different algorithms on the same data set. We showed that combining different algorithms and different representations, that all cover different aspects of the recommendation task, yields the best performance, confirming earlier work in the field of IR.

4. We have performed our experiments on publicly available data sets based on three different social bookmarking services covering two different domains of Web pages and scientific articles for a thorough evaluation of our work. This enhanced the generalizability of our findings.

5. We examined two problems associated with the growth of a social bookmarking websites: spam and duplicate content. We showed how prevalent these phenomena are. Moreover, we proposed methods for automatically detecting these phenomena, and examined the influence they might have on the item recommendation task.

9.4 Future Directions

In any research endeavor there is always room for improvement and the work described in this thesis is no different. While we have covered many different aspects of recommending items on social bookmarking websites in this thesis, we believe that the work we have done is but the tip of the proverbial iceberg. In particular, we acknowledge three fruitful and desired directions for future research.

User-based Evaluation We remarked already in the first chapter of this thesis that our choice for system-based evaluation—while necessary to whittle down the overwhelming number of possible algorithms and representations—leaves out perhaps the most important component of a recommender system: the user. Herlocker et al. (2004) was among the first to argue that user satisfaction is influenced by more than just recommendation accuracy, and McNee (2006) followed up on this work with extensive user studies of recommendation algorithms. Similarly, we believe it is essential to follow up our work with an evaluation with real users in realistic situations. Ideally, such experiments would have to be run in cooperation with one of the more popular social bookmarking websites to attract a large enough group of test subjects to be able to draw statistically significant conclusions about any differences in performance. Typically, such live user studies are done by performing so-called *A/B testing*, also known as randomized experiments or Control/Treatment testing (Kohavi et al., 2009). In A/B testing, two or more variants of a website are randomly assigned to the Web page visitors. With enough test subjects, meaningful conclusions can be drawn about, for instance, differences in clickthrough rates or purchases. To follow up on the work described in this thesis, it would be fruitful to compare different recommendation algorithms, such as the best variants of user-based filtering and item-based filtering, and the best content-based and hybrid filtering methods. Such user-based evaluation might see different algorithms rise to the top that were not the best-performing ones in the system-based experiments. For example, if users turn out to prefer serendipitous recommendations, we might see user-based CF as the algorithm with the highest clickthrough rate of the presented recommendations as suggested by McNee et al. (2006). Other influences on user satisfaction, and just as testable through A/B testing, could include the presences of explanations: why was a recommendation made. This has been shown to be important to user satisfaction as well (Herlocker et al., 2000).

Task Differentiation In our experiments we focused on one specific task: item recommendation based on all of the bookmarks or articles added by a user in the past. This is far from the only task that could be supported on a social bookmarking website. Figure 9.1 shows a matrix of possible tasks, each in the form of selecting an object type and then finding related objects, possibly of a different type, to go with them.

So far, related work on social tagging and social bookmarking has focused on 'Tag recommendation' and 'Guided search', and in this thesis we covered 'Item recommendation'. Other interesting and useful tasks still remain largely unexplored, such as finding like-minded users or selecting specific items to get recommendations for—commonly known as 'More like this' functionality—which could be very useful on a social bookmarking website. Some of our algorithms already perform some of these functions, such as finding similar users or

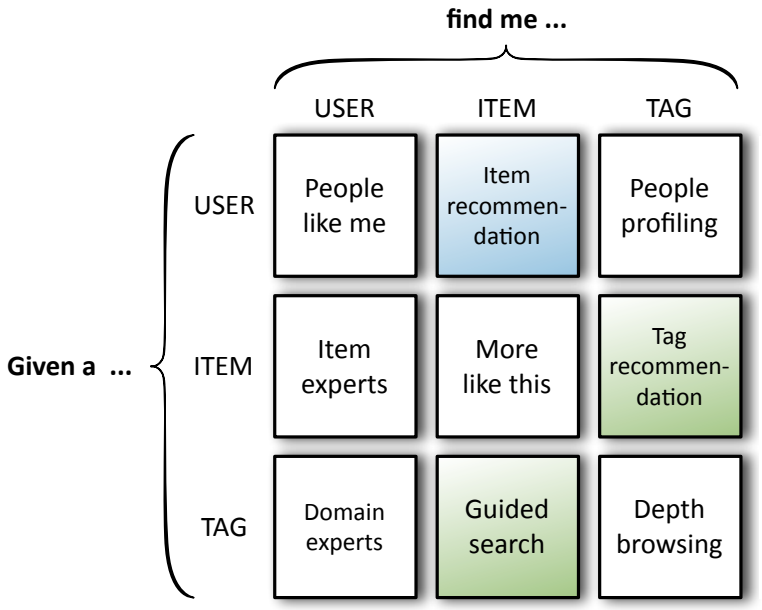


Figure 9.1: An overview of possible research tasks on social bookmarking websites, loosely adapted from Clements (2007). Most of the related work so far has focused on 'Tag recommendation', 'Guided search', and 'Item recommendation' (shaded cells). In this thesis we focused on the latter task.

items for the memory-based CF algorithms. However, more specialized algorithms might be better at this.

In deciding which tasks to tackle it is essential to profile the users: what do they want, how and what are they currently using the system for? We believe that focusing on real-world tasks in research can drive successful innovation, but only if the tasks under investigation are also desired by the users.

Algorithm Comparison In the experimental evaluation of our work, we have focused on using publicly available data sets and comparing our work against state-of-the-art approaches, something which is lacking from much of the related work. We are aware, however, that our comparisons are by no means complete as we picked only two promising approaches to compare our work with. Model-based collaborative filtering algorithms, for instance, were lacking from our comparison. In 'regular' recommendation experiments in different domains, model-based algorithms have been shown to hold a slight edge over memory-based algorithms, but without proper comparison on multiple social bookmarking data sets we cannot draw any conclusions about this. One natural extension of the work would therefore be to extend the comparison we made to all of the relevant related work discussed in Section 4.5. Such a comparison would have to include at least the approaches by Hotho et al. (2006a), Symeonidis et al. (2008b), Wetzker et al. (2009), Zanardi and Capra (2008), and Wang et al. (2006b).

References

Gediminas Adomavicius and Alexander Tuzhilin. 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, 2005.

Charu C. Aggarwal, Joel L. Wolf, Kun-Lung Wu, and Philip S. Yu. Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering. In *KDD '99: Proceeding of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 201–212, New York, NY, USA, 1999. ACM.

David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-Based Learning Algorithms. *Machine Learning*, 6(1):37–66, 1991.

Joshua Alspector, Aleksander Koicz, and Nachimuthu Karunanithi. Feature-based and Clique-based User Models for Movie Selection: A Comparative Study. *User Modeling and User-Adapted Interaction*, 7(4):279–304, 1997.

Sihem Amer-Yahia, Alban Galland, Julia Stoyanovich, and Cong Yu. From del.icio.us to x.qui.site: Recommendations in Social Tagging Sites. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1323–1326, New York, NY, USA, 2008. ACM.

Javed A. Aslam and Mark Montague. Models for Metasearch. In *SIGIR '01: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 276–284, New York, NY, USA, 2001. ACM.

Marco Balabanovic. *Learning to Surf: Multiagent Systems for Adaptive Web Page Recommendation*. PhD thesis, Stanford University, March 1998.

Shumeet Baluja, Rohan Seth, D. Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. Video Suggestion and Discovery for Youtube: Taking Random Walks through the View Graph. In *WWW '08: Proceedings of the 17th International Conference on World Wide Web*, pages 895–904, New York, NY, USA, 2008. ACM.

Shenghua Bao, Xiaoyuan Wu, Ben Fei, Guirong Xue, Zhong Su, and Yong Yu. Optimizing Web Search using Social Annotations. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pages 501–510, New York, NY, USA, 2007. ACM.

Brian T. Bartell, Garrison W. Cottrell, and Richard K. Belew. Automatic Combination of Multiple Ranked Retrieval Systems. In *SIGIR '94: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 173–181, New York, NY, 1994. Springer-Verlag New York, Inc.

Justin Basilico and Thomas Hofmann. Unifying Collaborative and Content-based Filtering. In *ICML '04: Proceedings of the 21st International Conference on Machine Learning*, pages 9–16, New York, NY, USA, 2004. ACM.

Chumki Basu, Haym Hirsh, and William W. Cohen. Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 714–720, 1998.

Chumki Basu, Haym Hirsh, William W. Cohen, and Craig Nevill-Manning. Technical Paper Recommendation: A Study in Combining Multiple Information Sources. *Journal of Artificial Intelligence Research*, 1:231–252, 2001.

Patrick Baudisch. Joining Collaborative and Content-based Filtering. In *Proceedings of the ACM CHI Workshop on Interacting with Recommender Systems*. ACM, May 1999.

Nicholas J. Belkin and W. Bruce Croft. Information Filtering and Information Retrieval: Two Sides of the Same Coin? *Communications of the ACM*, 35(12):29–38, 1992.

Nicholas J. Belkin, C. Cool, W. Bruce Croft, and James P. Callan. The Effect of Multiple Query Representations on Information Retrieval System Performance. In *SIGIR '93: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 339–346, New York, NY, USA, 1993. ACM.

Nicholas J. Belkin, P. Kantor, Edward A. Fox, and J.A. Shaw. Combining the Evidence of Multiple Query Representations for Information Retrieval. *Information Processing & Management*, 31(3): 431–448, May-June 1995.

Yaniv Bernstein and Justin Zobel. Redundant Documents and Search Effectiveness. In *CIKM '05: Proceedings of the Fourteenth International Conference on Information and Knowledge Management*, pages 736–743, New York, NY, USA, 2005. ACM.

Mikhail Bilenko, Raymond Mooney, William Cohen, Pradeep Ravikumar, and Stephen Fienberg. Adaptive Name Matching in Information Integration. *IEEE Intelligent Systems*, 18(5):16–23, 2003.

Kerstin Bischoff, Claudiu S. Firan, Wolfgang Nejdl, and Raluca Paiu. Can All Tags be used for Search? In *CIKM '08: Proceedings of the Seventeenth International Conference on Information and Knowledge Management*, pages 193–202, New York, NY, USA, 2008. ACM.

H.K. Biswas and Maruf Hasan. Using Publications and Domain Knowledge to Build Research Profiles: An Application in Automatic Reviewer Assignment. In *Proceedings of the 2007 International Conference on Information and Communication Technology (ICICT'07)*, pages 82–86, 2007.

Toine Bogers and Antal Van den Bosch. Comparing and Evaluating Information Retrieval Algorithms for News Recommendation. In *RecSys '07: Proceedings of the 2007 ACM Conference on Recommender Systems*, pages 141–144. ACM, October 2007.

Toine Bogers and Antal Van den Bosch. Recommending Scientific Articles using CiteULike. In *RecSys '08: Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 287–290. ACM, October 2008a.

Toine Bogers and Antal Van den Bosch. Using Language Models for Spam Detection in Social Bookmarking. In *Proceedings of 2008 ECML/PKDD Discovery Challenge Workshop*, pages 1–12, September 2008b.

Toine Bogers and Antal Van den Bosch. Using Language Modeling for Spam Detection in Social Reference Manager Websites. In Robin Aly, Claudia Hauff, I. den Hamer, Djoerd Hiemstra, Theo Huibers, and Franciska de Jong, editors, *Proceedings of the 9th Belgian-Dutch Information Retrieval Workshop (DIR 2009)*, pages 87–94, Enschede, February 2009a.

Toine Bogers and Antal Van den Bosch. Collaborative and Content-based Filtering for Item Recommendation on Social Bookmarking Websites. In Dietmar Jannach, Werner Geyer, Jill Freyne, Sarabjot Singh Anand, Casey Dugan, Bamshad Mobasher, and Alfred Kobsa, editors, *Proceedings of the ACM RecSys '09 workshop on Recommender Systems and the Social Web*, pages 9–16, October 2009b.

Kurt D. Bollacker, Steve Lawrence, and C. Lee Giles. Discovering Relevant Scientific Literature on the Web. *IEEE Intelligent Systems*, 15(2):42–47, 2000.

John S. Breese, David Heckerman, and Carl Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

Leo Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.

Eric Brill and Michele Banko. Scaling to Very Very Large Corpora for Natural Language Disambiguation. In *ACL '01: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 26–33, Morristown, NJ, USA, 2001. ACL.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John Lafferty, Robert L. Mercer, and Paul S. Roossin. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85, 1990.

Kenneth Bryan, Michael O'Mahony, and Pádraig Cunningham. Unsupervised Retrieval of Attack Profiles in Collaborative Recommender Systems. In *RecSys '08: Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 155–162, New York, NY, USA, 2008. ACM.

Jay Budzik and Kristian Hammond. Watson: Anticipating and Contextualizing Information Needs. In *Proceedings of the 62nd Annual Meeting of the American Society for Information Science*, Medford, NJ, 1999.

Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

Robin Burke. Integrating Knowledge-Based and Collaborative-Filtering Recommender Systems. In *Proceedings of the AAAI Workshop on AI in Electronic Commerce*, pages 69–72, 1999.

Robin Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

Robin Burke, Kristian J. Hammond, and Benjamin C. Young. The FindMe Approach to Assisted Browsing. *IEEE Expert: Intelligent Systems and Their Applications*, 12(4):32–40, 1997.

Vannevar Bush. As We May Think. *The Atlantic Monthly*, 176:101–108, July 1945.

John Canny. Collaborative Filtering with Privacy via Factor Analysis. In *SIGIR '02: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 238–245, New York, NY, USA, 2002. ACM.

A. Capocci and G. Caldarelli. Folksonomies and Clustering in the Collaborative System CiteULike. eprint arXiv: 0710.2835, 2007.

Mark J. Carman, Marik Baillie, and Fabio Crestani. Tag Data and Personalized Information Retrieval. In *SSM '08: Proceeding of the 2008 ACM Workshop on Search in Social Media*, pages 27–34, New York, NY, USA, 2008. ACM.

Ben Carterette, James Allan, and Ramesh Sitaraman. Minimal Test Collections for Retrieval Evaluation. In *SIGIR '06: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 268–275, New York, NY, USA, 2006. ACM.

Ciro Catutto, Christoph Schmitz, Andrea Baldassarri, Vito D. P. Servedio, Vittorio Loreto, Andreas Hotho, Miranda Grahl, and Gerd Stumme. Network Properties of Folksonomies. *AI Communications Journal, Special Issue on "Network Analysis in Natural Sciences and Engineering"*, 20:245–262, 2007.

Oscar Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2008.

Liren Chen and Katia Sycara. WebMate: A Personal Agent for Browsing and Searching. In *Proceedings of the Second International Conference on Autonomous Agents*, pages 132–139, New York, NY, 1998. ACM.

Paul-Alexandru Chirita, Claudi S. Firan, and Wolfgang Nejdl. Pushing Task Relevant Web Links down to the Desktop. In *Proceedings of the 8th ACM International Workshop on Web Information*

and Data Management (WIDM 2006), November 2006.

Paul-Alexandru Chirita, Stefania Costache, Wolfgang Nejdl, and Siegfried Handschuh. P-TAG: Large Scale Automatic Generation of Personalized Annotation Tags for the Web. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pages 845–854, New York, NY, USA, 2007. ACM.

Kimberley Chopin. Indexing, Managing, Navigating – A Micro- and Macro-examination of Folksonomic Tagging. Master's thesis, Royal School of Library and Information Science, Copenhagen, Denmark, August 2007.

Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. Combining Content-Based and Collaborative Filters in an Online Newspaper. In *Proceedings of ACM SIGIR Workshop on Recommender Systems*, August 1999.

Maarten Clements. Personalization of Social Media. In *Proceedings of the BCS IRSG Symposium: Future Directions in Information Access 2007*, August 2007.

Maarten Clements, Arjen P. de Vries, and Marcel J.T. Reinders. Optimizing Single Term Queries using a Personalized Markov Random Walk over the Social Graph. In *Proceedings of the ECIR Workshop on Exploiting Semantic Annotations in Information Retrieval (ESAIR '08)*, 2008a.

Maarten Clements, Arjen P. de Vries, and Marcel J.T. Reinders. Detecting synonyms in social tagging systems to improve content retrieval. In *SIGIR '08: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 739–740, New York, NY, USA, 2008b. ACM.

W. Bruce Croft. Combining Approaches to Information Retrieval. *Advances in Information Retrieval*, 7:1–36, 2000.

W. Bruce Croft and R.H. Thompson. I3R: A New Approach to the Design of Document Retrieval Systems. *Journal of the American Society for Information Science*, 38(6):389–404, 1987.

W. Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines: Information Retrieval in Practice*. Addison Wesley, 2009.

Abhinandan Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google News Personalization: Scalable Online Collaborative Filtering. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pages 271–280, 2007.

Marco De Gemmis, Pasquale Lops, Giovanni Semeraro, and Pierpaolo Basile. Integrating Tags in a Semantic Content-based Recommender. In *RecSys '08: Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 163–170, New York, NY, USA, 2008. ACM.

Michael De Nie. itList. *The Scout Report*, 6(18). Retrieved July 16, 2009, from http://www.mail-archive.com/scout-report@hypatia.cs.wisc.edu/msg00038.html, September 1999.

Brenda Dervin. From the Mind's Eye of the User: The Sense-Making Qualitative-Quantitative Methodology. *Qualitative Research in Information Management*, pages 61–84, 1992.

Susan T. Dumais and Jakob Nielsen. Automating the Assignment of Submitted Manuscripts to Reviewers. In *SIGIR '92: Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 233–244, New York, NY, USA, 1992. ACM.

Susan T. Dumais, Edward Cutrell, J.J. Cadiz, Gavin Jancke, Raman Sarin, and Daniel C. Robbins. Stuff I've Seen: a System for Personal Information Retrieval and Re-use. In *SIGIR '03: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 72–79, New York, NY, 2003. ACM.

Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, January 2007.

Stephen Farrell and Tessa Lau. Fringe Contacts: People-Tagging for the Enterprise. In *Proceedings of the WWW '06 Collaborative Web Tagging Workshop*, 2006.

Tom Fawcett. ROC Graphs: Notes and Practical Considerations for Researchers. *Machine Learning*, 31, 2004.

Ivan P. Fellegi and Alan B. Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, December 1969.

Martin Fenner. Interview with Kevin Emamy. Retrieved July 10, 2009, from http://network.nature.com/people/mfenner/blog/2009/01/30/interview-with-kevin-emamy, 2009.

S. Ferilli, N. Di Mauro, T.M.A. Basile, F. Esposito, and M. Biba. Automatic Topics Identification for Reviewer Assignment. *Advances in Applied Artificial Intelligence*, pages 721–730, 2006.

Edward A. Fox and Joseph A. Shaw. Combination of Multiple Searches. In *TREC-2 Working Notes*, pages 243–252, 1994.

Yoram Freund and Robert E. Schapire. Experiments with a New Boosting Algorithm. In L. Saitta, editor, *ICML '96: Proceedings of the 13th International Conference on Machine Learning*, pages 148–156, San Francisco, CA, 1996. Morgan Kaufmann.

Peter Garama and Patrick De Man. Finding Images Fast with Folksonomies. Master's thesis, Tilburg University, May 2008.

Natalie Glance, Jean-Luc Meunier, Pierre Bernard, and Damián Arregui. Collaborative Document Monitoring. In *GROUP '01: Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, pages 171–178, New York, NY, USA, 2001. ACM.

David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval*, 4(2):133–151, 2001.

Scott A. Golder and Bernardo A. Huberman. Usage Patterns of Collaborative Tagging Systems. *Journal of Information Science*, 32(2):198–208, 2006.

Anatoliy A. Gruzd and Michael B. Twidale. Write While You Search: Ambient Searching of a Digital Library in the Context of Writing. In *Proceedings of the 1st International Workshop on Digital Libraries in the Context of Users' Broader Activities (DL-CUBA), Joint Conference on Digital Libraries (JCDL'06)*, pages 13–16, 2006.

Zoltán Gyöngyi and Hector Garcia-Molina. Web Spam Taxonomy. In *AIRWeb '05: Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web*, pages 39–47, Chiba, Japan, May 2005.

Peter Haase, Marc Ehrig, Andreas Hotho, and Björn Schnizler. *Semantic Web and Peer-to-Peer*, chapter Personalized Information Access in a Bibliographic Peer-to-Peer System, pages 143–157. Springer Berlin Heidelberg, 2004.

Tony Hammond, Timo Hannay, Ben Lund, and Joanna Scott. Social Bookmarking Tools (I) - A General Review. *D-Lib Magazine*, 11(4):1082–9873, 2005.

Uri Hanani, Bracha Shapira, and Peretz Shoval. Information Filtering: Overview of Issues, Research and Systems. *User Modeling and User-Adapted Interaction*, 11:203–259, 2001.

David Hawking and Justin Zobel. Does Topic Metadata Help With Web Search? *Journal of the American Society for Information Science and Technology*, 58(5):613–628, 2007.

Monika Henzinger. Finding Near-duplicate Web Pages: A Large-scale Evaluation of Algorithms. In *SIGIR '06: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 284–291, New York, NY, USA, 2006. ACM.

Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An Algorithmic Framework for Performing Collaborative Filtering. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 230–237, New York, NY, USA, 1999. ACM.

Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining Collaborative Filtering Recommendations. In *CSCW '00: Proceedings of the 2000 ACM Conference on Computer Supported*

Cooperative Work, pages 241–250, New York, NY, USA, 2000. ACM.

Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

Mauricio A. Hernández and Salvatore J. Stolfo. The Merge/Purge Problem for Large Databases. In *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 127–138, New York, NY, USA, 1995. ACM.

Paul Heymann and Hector Garcia-Molina. Collaborative Creation of Communal Hierarchical Taxonomies in Social Tagging Systems. Technical report, Infolab, Stanford University, 2006.

Paul Heymann, Georgia Koutrika, and Hector Garcia-Molina. Fighting Spam on Social Web Sites: A Survey of Approaches and Future Challenges. *IEEE Internet Computing*, 11(6):36–45, 2007.

Paul Heymann, Georgia Koutrika, and Hector Garcia-Molina. Can Social Bookmarking Improve Web Search? In *WSDM '08: Proceedings of the International Conference on Web Search and Web Data Mining*, pages 195–206, New York, NY, USA, 2008a. ACM.

Paul Heymann, Daniel Ramage, and Hector Garcia-Molina. Social Tag Prediction. In *SIGIR '08: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 531–538, New York, NY, USA, July 2008b. ACM.

Djoerd Hiemstra. A Linguistically Motivated Probabilistic Model of Information Retrieval. In *ECDL '98: Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries*, pages 569—584, 1998.

Thomas Hofmann. Latent Semantic Models for Collaborative Filtering. *ACM Transactions on Information Systems*, 22(1):89–115, 2004.

Tzvetan Horozov, Nitya Narasimhan, and Venu Vasudevan. Using Location for Personalized POI Recommendations in Mobile Environments. In *SAINT '06: Proceedings of the International Symposium on Applications on Internet*, pages 124–129, Washington, DC, USA, 2006. IEEE Computer Society Press.

Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information Retrieval in Folksonomies: Search and Ranking. In *Proceedings of the European Semantic Web Conference 2006*, 2006a.

Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. BibSonomy: A Social Bookmark and Publication Sharing System. In *Proceedings of the Conceptual Structures Tool Interoperability Workshop at the 14th International Conference on Conceptual Structures*, pages 87–102, 2006b.

Andreas Hotho, Dominik Benz, Robert Jäschke, and Beate Krause. Introduction to the 2008 ECML/PKDD Discovery Challenge Workshop. In *Proceedings of 2008 ECML/PKDD Discovery Challenge Workshop*, September 2008.

Peter Ingwersen. Polyrepresentation of Information Needs and Semantic Entities: Elements of a Cognitive Theory for Information Retrieval Interaction. In *SIGIR '94: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 101–110, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

Peter Ingwersen. Cognitive Perspectives of Information Retrieval Interaction: Elements of a Cognitive IR Theory. *Journal of Documentation*, 52(1):3–50, 1996.

Peter Ingwersen. Context in Information Interaction – Revisited 2006. In Theo Bothma, editor, *Proceedings of the Fourth Biennial DISSAnet Conference*. University of Pretoria, November 2006.

Peter Ingwersen and Kalervo Järvelin. *The Turn: Integration of Information Seeking and Retrieval in Context*, volume 18 of *The Kluwer International Series on Information Retrieval*. Springer Verlag, Dordrecht, The Netherlands, 2005.

Anil K. Jain and D. Zongker. Feature Selection: Evaluation, Application, and Small Sample Performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.

Nathalie Japkowicz and Shaju Stephen. The Class Imbalance Problem: A Systematic Study. *Intelligent Data Analysis*, 6(5):429–449, 2002.

Robert Jäschke, Miranda Grahl, Andreas Hotho, Beate Krause, Christoph Schmitz, and Gerd Stumme. Organizing Publications and Bookmarks in BibSonomy. In Harith Alani, Natasha Noy, Gerd Stumme, Peter Mika, York Sure, and Denny Vrandecic, editors, *Proceedings of the WWW '07 Workshop on Social and Collaborative Construction of Structured Knowledge*, Banff, Canada, 2007a.

Robert Jäschke, Leandro Balby Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. Tag Recommendations in Folksonomies. In Joost N. Kok, Jacek Koronacki, Ramon López de Mántaras, Stan Matwin, Dunja Mladenic, and Andrzej Skowron, editors, *PKDD 2007: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 4702 of *Lecture Notes in Computer Science*, pages 506–514. Springer Verlag, 2007b.

Fredrick Jelinek. Self-organized Language Modeling for Speech Recognition. *Readings in Speech Recognition*, pages 450–506, 1990.

Thorsten Joachims, Dayne Freitag, and Tom Mitchell. WebWatcher: A Tour Guide for the World Wide Web. In *IJCAI '97: Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, volume 1, pages 770–777. Morgan Kaufmann, 1997.

Jaap Kamps and Maarten De Rijke. The Effectiveness of Combining Information Retrieval Strategies for European Languages. In *Proceedings 19th Annual ACM Symposium on Applied Computing*, pages 1073–1077, 2004.

George Karypis. Evaluation of Item-Based Top-N Recommendation Algorithms. In *CIKM '01: Proceedings of the Tenth International Conference on Information and Knowledge Management*, pages 247–254, New York, NY, USA, 2001. ACM.

Henry Kautz, Bart Selman, and Mehul Shah. Referral Web: Combining Social Networks and Collaborative Filtering. *Communications of the ACM*, 40(3):63–65, 1997.

Margaret E. I. Kipp. Complementary or Discrete Contexts in Online Indexing: A Comparison of User, Creator, and Intermediary Keywords. *Canadian Journal of Information and Library Science*, 2006.

Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal Henne. Controlled Experiments on the Web: Survey and Practical Guide. *Data Mining and Knowledge Discovery*, 18(1):140–181, 2009.

Yehuda Koren. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *KDD '08: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434, New York, NY, USA, 2008. ACM.

Georgia Koutrika, Frans Adjie Effendi, Zoltán Gyöngyi, Paul Heymann, and Hector Garcia-Molina. Combating Spam in Tagging Systems. In *AIRWeb '07: Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web*, pages 57–64, New York, NY, USA, 2007. ACM.

Beate Krause, Andreas Hotho, and Gerd Stumme. The Anti-Social Tagger - Detecting Spam in Social Bookmarking Systems. In *AIRWeb '08: Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web*, 2008.

Ralf Krestel and Ling Chen. Using Co-occurrence of Tags and Resources to Identify Spammers. In *Proceedings of 2008 ECML/PKDD Discovery Challenge Workshop*, pages 38–46, September 2008.

Shyong K. Lam and John Riedl. Shilling Recommender Systems for Fun and Profit. In *WWW '04: Proceedings of the 13th International Conference on World Wide Web*, pages 393–402, New York, NY, USA, 2004. ACM.

Renaud Lambiotte and Marcel Ausloos. Collaborative Tagging as a Tripartite Network. *Lecture Notes in Computer Science*, 3993(2006):1114–1117, 2006.

F. W. Lancaster. *Indexing and Abstracting in Theory and Practice*. University of Illinois, Graduate School of Library and Information, 3rd edition, 2003.

Ken Lang. NewsWeeder: Learning to Filter Netnews. In *ICML '95: Proceedings of the 12th International Conference on Machine Learning*, pages 331–339, San Mateo, CA, USA, 1995. Morgan Kaufmann.

Birger Larsen, Peter Ingwersen, and Berit Lund. Data Fusion According to the Principle of Polyrepresentation. *Journal of the American Society for Information Science and Technology*, 60(4):646–654, 2009.

Liz Lawley. Tagging vs. Folksonomy. Retrieved July 17, 2009, from http://many.corante.com/archives/2006/11/02/tagging_vs_folksonomy.php, November 2006.

Julia Lawlor. Web Services Offer Solutions to Bookmark Overload. Retrieved July 16, 2009, from http://www.nytimes.com/2000/07/13/technology/web-services-offer-solutions-to-bookmark-overload.html, July 2000.

Joon Ho Lee. Analyses of Multiple Evidence Combination. *SIGIR Forum*, 31(SI):267–276, 1997.

Rui Li, Shenghua Bao, Ben Fei, Zhong Su, and Yong Yu. Towards Effective Browsing of Large Scale Social Annotations. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pages 943–951, 2007.

Henry Lieberman. Letizia: An Agent That Assists Web Browsing. In Chris S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 924–929, San Mateo, CA, 1995. Morgan Kaufmann.

Greg Linden, Brent Smith, and Jeremy York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

Veronica Maidel, Peretz Shoval, Bracha Shapira, and Meirav Taieb-Maimon. Evaluation of an Ontology-Content Based Filtering Method for a Personalized Newspaper. In *RecSys '08: Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 91–98, New York, NY, USA, 2008. ACM.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

Cameron Marlow, Mor Naaman, Danah Boyd, and Marc Davis. HT06, tagging paper, taxonomy, Flickr, academic article, to read. In *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 31–40, New York, NY, USA, 2006. ACM.

Adam Mathes. Folksonomies - Cooperative Classification and Communication through Shared Metadata. Technical report, University of Illinois Urbana-Champaign, 2004.

Sean McNee. *Meeting User Information Needs in Recommender Systems*. PhD thesis, University of Minnesota, June 2006.

Sean McNee, Istvan Albert, Dan Cosley, Prateep Gopalkrishnan, Shyong K. Lam, Al Mamunur Rashid, Joseph A. Konstan, and John Riedl. On the Recommending of Citations for Research Papers. In *CSCW '02: Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work*, pages 116–125, New York, NY, USA, 2002. ACM.

Sean McNee, Nishikant Kapoor, and Joseph A. Konstan. Don't Look Stupid: Avoiding Pitfalls when Recommending Research Papers. In *CSCW '06: Proceedings of the 2006 ACM Conference on Computer Supported Cooperative Work*, pages 171–180, New York, NY, USA, 2006. ACM.

Bhashkar Mehta. *Cross-System Personalization: Enabling Personalization across Multiple Systems*. PhD thesis, Universität Duisberg Essen, 2008.

Peter Mika. Ontologies are us: A Unified Model of Social Networks and Semantics. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *ISWC 2005*, volume 3729 of *Lecture Notes on Computer Science*, pages 522–536, Berlin, November 2005. Springer-Verlag.

Gilad Mishne. AutoTag: A Collaborative Approach to Automated Tag Assignment for Weblog Posts. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, May 2006.

Gilad Mishne and Maarten de Rijke. Deriving Wishlists from Blogs - Show us your Blog, and We'll Tell you What Books to Buy. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, pages 925–926, New York, NY, 2006. ACM.

Gilad Mishne, David Carmel, and Ronny Lempel. Blocking Blog Spam with Language Model Disagreement. In *AIRWeb '05: Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web*, pages 1–6, New York, NY, USA, 2005. ACM.

Bamshad Mobasher, Robin Burke, Chad Williams, and Runa Bhaumik. Analysis and Detection of Segment-focused Attacks against Collaborative Recommendation. In Olfa Nasraoui, Osmar R. Zaïane, Myra Spiliopoulou, Bamshad Mobasher, Brij Masand, and Philip S. Yu, editors, *Web Mining and Web Usage Analysis*, volume 4198 of *Lecture Notes in Computer Science*, pages 96–118. Springer, 2006.

Miquel Montaner, Beatriz López, and Josep Lluís de la Rosa. A Taxonomy of Recommender Agents on the Internet. *Artificial Intelligence Review*, 19(4):285–330, 2003.

Raymond J. Mooney and Loriene Roy. Content-Based Book Recommending Using Learning for Text Categorization. In *DL '00: Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 195–204, New York, NY, 2000. ACM.

P. Jason Morrison. Tagging and Searching: Search Retrieval Effectiveness of Folksonomies on the Web. Master's thesis, Kent State University, May 2007.

Reyn Nakamoto, Shinsuke Nakajima, Jun Miyazaki, and Shunsuke Uemura. Tag-Based Contextual Collaborative Filtering. In *Proceedings of the 18th IEICE Data Engineering Workshop*, 2007.

Reyn Nakamoto, Shinsuke Nakajima, Jun Miyazaki, Shunsuke Uemura, Hirokazu Kato, and Yoichi Inagaki. Reasonable Tag-based Collaborative Filtering for Social Tagging Systems. In *WICOW '08: Proceeding of the 2nd ACM Workshop on Information Credibility on the Web*, pages 11–18, New York, NY, USA, 2008. ACM.

Gonzalo Navarro. A Guided Tour to Approximate String Matching. *ACM Computing Surveys*, 33(1): 31–88, 2001.

Michael O'Mahony, Neil Hurley, Nicholas Kushmerick, and Guénolé Silvestre. Collaborative Recommendation: A Robustness Analysis. *ACM Transactions on Internet Technology (TOIT)*, 4(4): 344–377, 2004.

Tim O'Reilly. What Is Web 2.0. Retrieved July 7, 2009, from http://oreilly.com/web2/archive/what-is-web-20.html, 2005.

Iadh Ounis, Maarten de Rijke, Craig McDonald, Gilad Mishne, and Ian Soboroff. Overview of the TREC 2006 Blog Track. In *TREC 2006 Working Notes*, 2006.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

Parag and Pedro Domingos. Multi-Relational Record Linkage. In *Proceedings of the KDD-2004 Workshop on Multi-Relational Data Mining*, pages 31–48. ACM, 2004.

Patrick Paulson and Aimillia Tzanavari. Combining Collaborative and Content-Based Filtering Using Conceptual Graphs. *Modelling with Words*, pages 168–185, 2003.

Michael J. Pazzani. A Framework for Collaborative, Content-based and Demographic Filtering. *Artificial Intelligence Review*, 13(5):393–408, 1999.

Michael J. Pazzani, Jack Muramatsu, and Daniel Billsus. Syskill & Webert: Identifying Interesting Web Sites. In *AAAI/IAAI, Vol. 1*, pages 54–61, 1996.

Peter Pirolli and Stuart Card. Information Foraging in Information Access Environments. In *CHI '95: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 51–58, New York, NY, 1995. ACM.

Jay M. Ponte and W. Bruce Croft. A Language Modeling Approach to Information Retrieval. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, New York, NY, 1998. ACM.

Reid Priedhorsky, Jilin Chen, Shyong K. Lam, Katherine Panciera, Loren Terveen, and John Riedl. Creating, Destroying, and Restoring Value in Wikipedia. In *Proceedings of GROUP '07*, 2007.

Mari Carmen Puerta Melguizo, Olga Muñoz Ramos, Lou Boves, Toine Bogers, and Antal Van den Bosch. A Personalized Recommender System for Writing in the Internet Age. In *Proceedings of the LREC 2008 workshop on Natural Language Processing Resources, Algorithms, and Tools for Authoring Aids*, 2008.

Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean McNee, Joseph A. Konstan, and John Riedl. Getting to Know You: Learning New User Preferences in Recommender Systems. In *IUI '02: Proceedings of the 7th International Conference on Intelligent User Interfaces*, pages 127–134, New York, NY, 2002. ACM.

M. Elena Renda and Umberto Straccia. Web Metasearch: Rank vs. Score-based Rank Aggregation Methods. In *SAC '03: Proceedings of the 2003 ACM Symposium on Applied Computing*, pages 841–846, New York, NY, USA, 2003. ACM.

Paul Resnick and Rahul Sami. The Influence Limiter: Provably Manipulation-Resistant Recommender Systems. In *RecSys '07: Proceedings of the 2007 ACM Conference on Recommender Systems*, pages 25–32, New York, NY, USA, 2007. ACM.

Paul Resnick and Hal R. Varian. Recommender Systems. *Communications of the ACM*, 40(3):56–58, 1997.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *CSCW '94: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pages 175–186, New York, NY, USA, 1994. ACM.

Bradley J. Rhodes and Patti Maes. Just-In-Time Information Retrieval Agents. *IMB Systems Journal*, 39(3-4):685–704, 2000.

Stuart J. Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Upper Saddle River, NJ, 2nd edition, 2003.

Gerard Salton and Christopher Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-Based Collaborative Filtering Recommendation Algorithms. In *WWW '01: Proceedings of the 10th International Conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

Badrul Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Incremental SVD-Based Algorithms for Highly Scaleable Recommender Systems. In *Proceedings of the Fifth International Conference on Computer and Information Technology (ICCIT 2002)*, 2002.

Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and Metrics for Cold-start Recommendations. In *SIGIR '02: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253–260, New York, NY, USA, 2002. ACM.

S. Schmitt and R. Bergmann. Applying Case-based Reasoning Technology for Product Selection and Customization in Electronic Commerce Environments. In *Proceedings of the 12th Bled Electronic Commerce Conference*, Bled, Slovenia, 1999.

Christoph Schmitz, Andreas Hotho, Robert Jäschke, and Gerd Stumme. Mining Association Rules in Folksonomies. In V. Batagelj, H.-H. Bock, A. Ferligoj, and A. Žiberna, editors, *Data Science and Classification: Proceedings of the 10th IFCS Conference*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 261–270, Heidelberg, July 2006. Springer-Verlag.

Christoph Schmitz, Miranda Grahl, Andreas Hotho, Gerd Stumme, Ciro Cattuto, Andrea Baldassarri, Vittorio Loreto, and Vito D. P. Servedio. Network Properties of Folksonomies. In Scott Golder and Frank Smadja, editors, *Proceedings of the WWW '07 Tagging and Metadata for Social Information Organization Workshop*, 2007.

Stephen B. Seidman. Network Structure and Minimum Degree. *Social Networks*, 5(5):269–287, 1983.

Upendra Shardanand and Pattie Maes. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *CHI '95: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 210–217, New York, NY, USA, 1995. ACM.

Prashant Sharma. Core Characteristics of Web 2.0 Services. Retrieved July 7, 2009, from http://www.techpluto.com/web-20-services/, 2008.

Kaikai Shen and Lide Wu. Folksonomy as a Complex Network. Technical report, Department of Computer Science, Fudan University, 2005.

Clay Shirky. Ontology is Overrated: Categories, Links, and Tags. Retrieved July 17, 2009, from http://www.shirky.com/writings/ontology_overrated.html, 2005.

Börkur Sigurbjörnsson and Roelof Van Zwol. Flickr Tag Recommendation based on Collective Knowledge. In *WWW '08: Proceedings of the 17th International Conference on World Wide Web*, pages 327–336, New York, NY, USA, 2008. ACM.

Mette Skov, Birger Larsen, and Peter Ingwersen. Inter and Intra-Document Contexts Applied in Polyrepresentation for Best Match IR. *Information Processing & Management*, 44:1673–1683, 2008.

Yang Song, Ziming Zhuang, Huajing Li, Qiankun Zhao, Jia Li, Wang-Chien Lee, and C. Lee Giles. Real-time Automatic Tag Recommendation. In *SIGIR '08: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 515–522, New York, NY, USA, 2008. ACM.

Anselm Spoerri. Authority and Ranking Effects in Data Fusion. *Journal of the American Society for Information Science and Technology*, 59(3):450–460, 2007.

Lubomira Stoilova, Todd Holloway, Ben Markines, Ana G. Maguitman, and Filippo Menczer. GiveALink: Mining a Semantic Network of Bookmarks for Web Search and Recommendation. In *LinkKDD '05: Proceedings of the 3rd International Workshop on Link Discovery*, pages 66–73, New York, NY, USA, 2005. ACM.

Trevor Strohman, Donald Metzler, and W. Bruce Croft. Indri: A Language Model-based Search Engine for Complex Queries. In *Proceedings of the International Conference on Intelligence Analysis*, May 2005.

Trevor Strohman, W. Bruce Croft, and David Jensen. Recommending Citations for Academic Papers. In *SIGIR '07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 705–706, New York, NY, 2007. ACM.

Fabian M. Suchanek, Milan Vojnovic, and Dinan Gunawardena. Social Tags: Meaning and Suggestions. In *CIKM '08: Proceedings of the Seventeenth International Conference on Information and Knowledge Management*, pages 223–232, New York, NY, USA, 2008. ACM.

Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. Tag Recommendations Based on Tensor Dimensionality Reduction. In *RecSys '08: Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 43–50, New York, NY, USA, 2008a. ACM.

Panagiotis Symeonidis, Maria Ruxanda, Alexandros Nanopoulos, and Yannis Manolopoulos. Ternary Semantic Analysis of Social Tags for Personalized Music Recommendation. In *ISMIR '08: Proceedings of the 9th International Conference on Music Information Retrieval*, pages 219–224, 2008b.

Martin Szomszor, Ciro Cattuto, Harith Alani, Kieron O'Hara, Andrea Baldassarri, Vittorio Loreto, and Vito D.P. Servedio. Folksonomies, the Semantic Web, and Movie Recommendation. In *Proceedings of the ESWC Workshop on Bridging the Gap between Semantic Web and Web 2.0*, 2007.

Robert S. Taylor. Question-Negotiation and Information Seeking in Libraries. *College and Research Libraries*, 29:178–194, May 1968.

Nava Tintarev and Judith Masthoff. Similarity for News Recommender Systems. In *Proceedings of the AH'06 Workshop on Recommender Systems and Intelligent User Interfaces*, 2006.

B. Towle and C. Quinn. Knowledge Based Recommender Systems Using Explicit User Models. In *Proceedings of the AAAI Workshop on Knowledge-Based Electronic Markets*, AAAI Technical Report WS-00-04, pages 74–77, Menlo Park, CA, 2000. AAAI Press.

Karen H. L. Tso-Sutter, Leandro Balby Marinho, and Lars Schmidt-Thieme. Tag-aware Recommender Systems by Fusion of Collaborative Filtering Algorithms. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 1995–1999, New York, NY, USA, 2008. ACM.

Antal Van den Bosch. Wrapped Progressive Sampling Search for Optimizing Learning Algorithm Parameters. In R. Verbrugge, N. Taatgen, and L. Schomaker, editors, *Proceedings of the 16th Belgian-Dutch Conference on Artificial Intelligence*, pages 219–226, 2004.

Mark Van Setten, Mettina Veenstra, Anton Nijholt, and Betsy van Dijk. Case-Based Reasoning as a Prediction Strategy for Hybrid Recommender Systems. In *Proceedings of Advances in Web Intelligence: Second International Atlantic Web Intelligence Conference (AWIC 2004)*, volume 3034 of *Lecture Notes in Computer Science*, pages 13–22, Berlin, 2004. Springer Verlag.

Thomas Vander Wal. Explaining and Showing Broad and Narrow Folksonomies. Retrieved April 29, 2008, from http://www.personalinfocloud.com/2005/02/explaining_and_.html, 2005a.

Thomas Vander Wal. Folksonomy Definition and Wikipedia. Retrieved July 17, 2009, from http://www.vanderwal.net/random/entrysel.php?blog=1750, November 2005b.

Christopher C. Vogt and Garrison W. Cottrell. Predicting the Performance of Linearly Combined IR Systems. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 190–196, New York, NY, 1998. ACM.

Luis Von Ahn, Ben Maurer, Colin McMillen, David Abraham, and Manuel Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science*, 321:1465–1468, September 2008.

Ellen M. Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. Learning Collection Fusion Strategies. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *SIGIR '95: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–179, New York, NY, 1995. ACM.

Jakob Voß. Tagging, Folksonomy & Co–Renaissance of Manual Indexing? In *ISI 2007: Proceedings of the 10th International Symposium for Information Science*, 2007.

Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion. In *SIGIR '06: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 501–508, New York, NY, USA, 2006a. ACM.

Xuanhui Wang, Jian-Tao Sun, Zheng Chen, and Chengxiang Zhai. Latent Semantic Analysis for Multiple-Type Interrelated Data Objects. In *SIGIR '06: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 236–243, New York, NY, USA, 2006b. ACM.

Duncan J. Watts and Steven H. Strogatz. Collective Dynamics of 'Small-world' Networks. *Nature*, 393(6684):440–442, June 1998.

Sholom M. Weiss and Casimir A. Kulikowski. *Computer Systems That Learn*. Morgan Kaufmann, 1991.

Robert Wetzker. Analyzing Social Bookmarking Systems: A del.icio.us Cookbook. In *Proceedings of the ECAI 2008 Mining Social Data Workshop*, pages 26–30. IOS Press, July 2008.

Robert Wetzker, Winfried Umbrath, and Alan Said. A Hybrid Approach to Item Recommendation in Folksonomies. In *ESAIR '09: Proceedings of the WSDM '09 Workshop on Exploiting Semantic Annotations in Information Retrieval*, pages 25–29, New York, NY, USA, 2009. ACM.

Brian Whitman and Steve Lawrence. Inferring Descriptions and Similarity for Music from Community Metadata. In *Proceedings of the 2002 International Computer Music Conference*, pages 591–598, 2002.

David H. Wolpert. Stacked Generalization. *Neural Networks*, 5(2):241–259, 1992.

Zhichen Xu, Yun Fu, Jianchang Mao, and Difu Su. Towards the Semantic Web: Collaborative Tag Suggestions. In *Proceedings of the WWW '06 Collaborative Web Tagging Workshop*, 2006.

Yusuke Yanbe, Adam Jatowt, Satoshi Nakamura, and Katsumi Tanaka. Can Social Bookmarking Enhance Search in the Web? In *JCDL '07: Proceedings of the 7th ACM/IEEE Joint Conference on Digital Libraries*, pages 107–116, New York, NY, USA, 2007. ACM.

David Yarowsky and Radu Florian. Taking the Load off the Conference Chairs: Towards a Digital Paper-Routing Assistant. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in NLP and Very Large Corpora*, pages 220–230, 1999.

Hilmi Yildirim and Mukkai S. Krishnamoorthy. A Random Walk Method for Alleviating the Sparsity Problem in Collaborative Filtering. In *RecSys '08: Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 131–138, New York, NY, USA, 2008. ACM.

Ping Yin, Ming Zhang, and Xiaoming Li. Recommending Scientific Literatures in a Collaborative Tagging Environment. *Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers*, pages 478–481, 2007.

Valentina Zanardi and Licia Capra. Social Ranking: Uncovering Relevant Content using Tag-based Recommender Systems. In *RecSys '08: Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 51–58, New York, NY, USA, 2008. ACM.

Chengxiang Zhai and John Lafferty. A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *ACM Transactions on Information Systems*, 22(2):179–214, 2004.

Ding Zhou, Jiang Bian, Shuyi Zheng, Hongyuan Zha, and C. Lee Giles. Exploring Social Annotations for Information Retrieval. In *WWW '08: Proceedings of the 17th International Conference on World Wide Web*, pages 715–724, New York, NY, USA, 2008. ACM.

Cai-Nicolas Ziegler, Georg Lausen, and Lars Schmidt-Thieme. Taxonomy-driven Computation of Product Recommendations. In *CIKM '04: Proceedings of the Thirteenth International Conference on Information and Knowledge Management*, pages 406–415, 2004.

COLLECTING THE CITEULIKE DATA SET

A.1 Extending the Public Data Dump

We described in Chapter 3 that our CiteULike data set is based on the November 2, 2007 data dump that is made available publicly[1] by CiteULike. This dump contains all information on which articles were posted by whom, with which tags, and at what point in time. Figure A.1 shows a tiny subset of this data dump.

ARTICLE ID	USER ID (HASHED)	TIMESTAMP	TAG
42	217369743f9df99964fa16439a01f5f3	2006-12-26T02:03:08.605006+00	metabolism
42	217369743f9df99964fa16439a01f5f3	2006-12-26T02:03:08.605006+00	networks
42	217369743f9df99964fa16439a01f5f3	2006-12-26T02:03:08.605006+00	systems_biology
42	473d9d0924d3aafcf26846d7772a0ae5	2005-06-24T01:22:12.696806+01	small_world
42	473d9d0924d3aafcf26846d7772a0ae5	2005-06-24T01:22:12.696806+01	wireing_digrams
42	61baaeba8de136d9c1aa9c18ec3860e8	2004-11-04T02:25:05.373798+00	barabasi
42	61baaeba8de136d9c1aa9c18ec3860e8	2004-11-04T02:25:05.373798+00	ecoli
42	61baaeba8de136d9c1aa9c18ec3860e8	2004-11-04T02:25:05.373798+00	metabolism
42	61baaeba8de136d9c1aa9c18ec3860e8	2004-11-04T02:25:05.373798+00	networks
42	b8f7de09e5c78bdd72b2fd9c4bb00fc2	2006-06-02T18:02:00.510405+01	metabolism
42	b8f7de09e5c78bdd72b2fd9c4bb00fc2	2006-06-02T18:02:00.510405+01	network
43	951c3216b53eeb01d5ddc75ecfe63753	2007-05-11T16:47:11.366525+01	no-tag
43	32c2ddd35515c3fdc1b7731992887015	2007-01-16T19:45:20.535288+00	key--reg ulatory_network
43	32c2ddd35515c3fdc1b7731992887015	2007-01-16T19:45:20.535288+00	key--robustness
43	32c2ddd35515c3fdc1b7731992887015	2007-01-16T19:45:20.535288+00	key--systems_biology
43	32c2ddd35515c3fdc1b7731992887015	2007-01-16T19:45:20.535288+00	type--review
43	86c091c8b7d793966136c07488efe622	2006-12-01T16:02:59.704955+00	robustness

Figure A.1: A small subset of a CiteULike data dump. The columns from left to right contain the article IDs, user IDs. time stamps, and tags.

Each line represents a user-item-tag triple with the associated timestamp of the posting, so if a user posted an article with n tags, then this resulted in n rows in the file for that article-user pair. If a user added no tags, then the article-user pair is represented by 1 row with the tag no-tag. On the CiteULike website, users can pick their own user name, but these

[1] Available from http://www.citeulike.org/faq/data.adp.

were hashed in the data dumps using the MD5 cryptographic hashing function for privacy reasons. Unfortunately, most of the interesting features on the CiteULike website are linked to the user names and not to the hashed IDs. For instance, each article has a separate page for the users that added it, which contain the personal metadata such as reviews and comments. Unfortunately, these article pages display only the CiteULike user names, which means that for us to be able to link the data dump to the personal metadata we had to match each user name to its corresponding MD5 hashes.

Directly applying the MD5 hashing function to the username did not result in any matching hashes, which suggests that CiteULike uses a *salt* to make reverse lookup practically impossible[2]. We therefore had to find another way of matching the user names to the hashes. First, we crawled all user-specific article pages. Each article on CiteULike has its own generic page, which can be accessed by inserting the article ID from the CiteULike data dump in a URL of the form `http://www.citeulike.org/article/`**ARTICLE_ID**. Each article page contains metadata and also lists, by user name, all users that have added the article to their personal library. We crawled all article pages and collected all information. After acquiring the names of the users that added an article, we also crawled all user-specific versions of these pages. These can be accessed at a URL of the form `http://www.citeulike.org/user/`**USER_NAME**`/article/`**ARTICLE_ID**. Using the collected lists of user names, we collected the personal metadata added by each separate user such as tags and comments.

By collecting these article-user name-tag triples from the CiteULike article pages, we were then able to match user names to MD5 hashes[3]. This turned the alignment problem into a matter of simple constraint satisfaction. For a single article *A* there might be multiple authors who added the same tag(s) to *A*, but perhaps only one of those authors added another article *B* with his own unique tags. By identifying those posts that matched uniquely on articles and tags, we were able to align that user's name with the hashed ID. We ran this alignment process for 7 iterations, each round resolving more ambiguous matches, until no more authors could be matched. Table A.1 shows statistics of our progress in aligning the usernames and hashes.

Table A.1: Statistics of the user alignment process. Each cell lists how many users and associated articles and tags were matched after the seven steps.

	Data dump	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7
Users	27,133	24,591	25,182	25,260	25,284	25,292	25,352	25,375
Articles	813,186	763,999	780,244	780,837	780,851	780,851	803,278	803,521
Tags	235,199	224,061	227,678	227,844	227,848	227,848	232,794	232,837

After seven rounds, 93.5% of the initial 27,133 user names were correctly identified. Furthermore, 98.8% of all articles were retained and 99.0% of all tags. We believe that this represents a substantial enough subset of our the original data dump to proceed with further

[2] A *salt* is a series of random bits used as additional input to the hashing function. It makes reverse lookup using dictionary attacks more complicated by effectively extending the length and potentially the complexity of the password.

[3] Matching usernames and hashes on the specific timestamps when articles were added would have been easier and more specific, but this was not possible: the timestamps were not present on the crawled Web pages.

experiments. Figure A.2 shows the same extract of the CiteULike data dump of Figure A.1 with the resolved author names. For instance, we were able to resolve the user ID hash `61baaeba8de136d9c1aa9c18ec3860e8` to the user name `camster`.

ARTICLE ID	USER NAME	TIMESTAMP	TAG
42	alexg	2006-12-26T02:03:08.605006+00	metabolism
42	alexg	2006-12-26T02:03:08.605006+00	networks
42	alexg	2006-12-26T02:03:08.605006+00	systems_biology
42	barry	2005-06-24T01:22:12.696806+01	small_world
42	barry	2005-06-24T01:22:12.696806+01	wireing_digrams
42	camster	2004-11-04T02:25:05.373798+00	barabasi
42	camster	2004-11-04T02:25:05.373798+00	ecoli
42	camster	2004-11-04T02:25:05.373798+00	metabolism
42	camster	2004-11-04T02:25:05.373798+00	networks
42	tny	2006-06-02T18:02:00.510405+01	metabolism
42	tny	2006-06-02T18:02:00.510405+01	network
43	OriginalLurch	2007-05-11T16:47:11.366525+01	no-tag
43	adepace	2007-01-16T19:45:20.535288+00	key--regulatory_network
43	adepace	2007-01-16T19:45:20.535288+00	key--robustness
43	adepace	2007-01-16T19:45:20.535288+00	key--systems_biology
43	adepace	2007-01-16T19:45:20.535288+00	type--review
43	anatiferous	2006-12-01T16:02:59.704955+00	robustness

Figure A.2: The same small subset of the CiteULike data dump from Figure A.1 after aligning the user names with the hashed IDs. The columns from left to right contain the article IDs, user names. time stamps, and tags.

A.2 Spam Annotation

Figure A.3 illustrates the straightforward interface we created for the spam annotation process. For each user it randomly selects a maximum of five articles and displays the article title (if available) and the associated tags. It also shows a link to the CiteULike page of the article. Preliminary analysis showed that articles that were clearly spam were usually already removed by CiteULike and returned a *404 Not Found* error. We therefore instructed our judges to check the CiteULike links if a user's spam status was not obvious from the displayed articles. Missing article pages meant users should be marked as spam. In this process, we assumed that although spam users might add real articles to their profile in an attempt to evade detection, real dedicated CiteULike users would never willingly add spam articles to their profile. Finally, we noticed that spam content was injected into CiteULike in many different languages. From the experience of the annotators, most spam was in English, but considerable portions were in Spanish, Swedish, and German. Other languages in which spam content was found were, among others, Dutch, Finnish, Chinese, and Italian.

Of the 5,200 users in our subset, 3,725 (or 28.4%) were spam users, which is a much smaller proportion than in the BibSonomy system, but still a considerable part of the entire data set. The numbers in Table 7.1 are reported for this 20% sample of CiteULike users. It is likely that such a sizable chunk of spam can have a significant effect on the recommendation performance. We test this hypothesis about the influence of spam in Section 7.4.

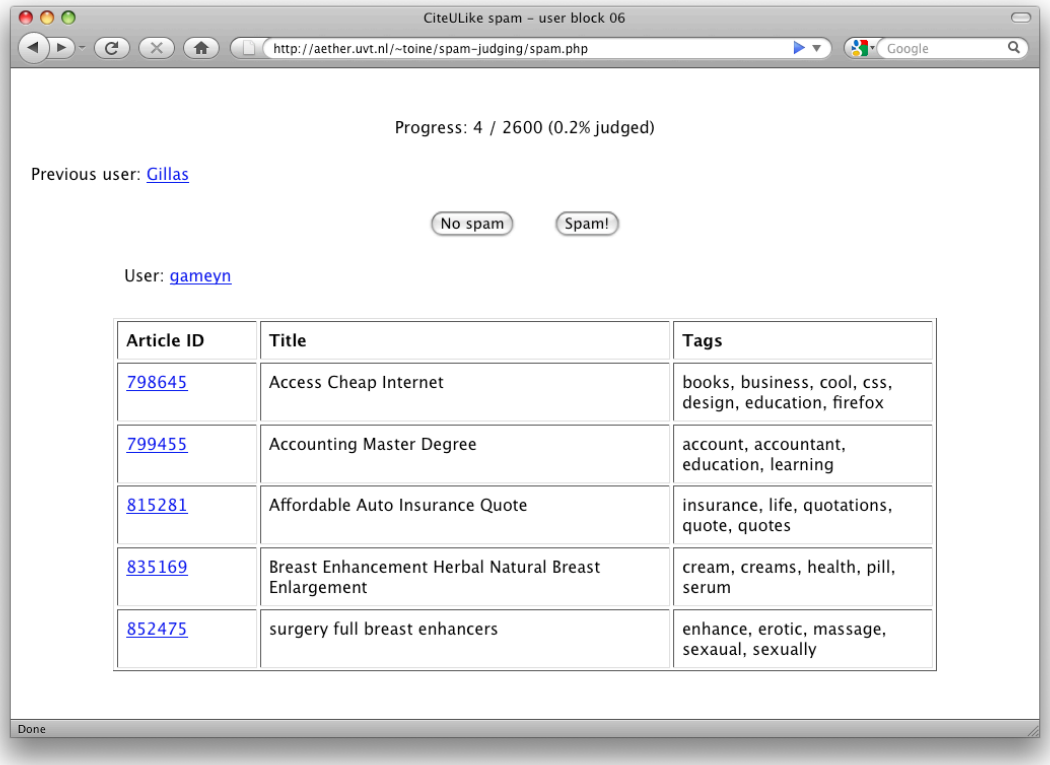


Figure A.3: A screenshot of the interface used to annotate a subset of CiteULike users as possible spam users. The user in the screenshot is a typical example of a spammer. For every user, we display up to five randomly selected posts with the article title and the assigned tags.

APPENDIX B

GLOSSARY OF RECOMMENDATION RUNS

In this appendix we list the run names used in Chapter 4 with a brief explanation for the convenience of the reader in Tables B.1 and B.2.

Table B.1: Run names used in Chapter 4.

Run name	Run description
U-BIN-SIM	User-based CF using binary vectors from the ratings matrix **R** and cosine similarity (Subsection 4.3.1).
U-BIN-IDF-SIM	User-based CF using tf·idf-weighted vectors from the ratings matrix **R** and cosine similarity (Subsection 4.3.1).
I-BIN-SIM	Item-based CF using binary vectors from the ratings matrix **R** and cosine similarity (Subsection 4.3.1).
I-BIN-IDF-SIM	Item-based CF using tf·idf-weighted vectors from the ratings matrix **R** and cosine similarity (Subsection 4.3.1).
UT-JACCARD-SIM	User-based CF using binary vectors from the **UT** matrix and Jaccard overlap (Subsection 4.4.1).
UT-DICE-SIM	User-based CF using binary vectors from the **UT** matrix and Dice's coefficient (Subsection 4.4.1).
UT-BIN-SIM	User-based CF using binary vectors from the **UT** matrix and cosine similarity (Subsection 4.4.1).
UT-TF-SIM	User-based CF using tag frequency vectors from the **UT** matrix and cosine similarity (Subsection 4.4.1).
UT-TFIDF-SIM	User-based CF using tf·idf-weighted tag frequency vectors from the **UT** matrix and cosine similarity (Subsection 4.4.1).
IT-JACCARD-SIM	Item-based CF using binary vectors from the **IT** matrix and Jaccard overlap (Subsection 4.4.1).
IT-DICE-SIM	Item-based CF using binary vectors from the **IT** matrix and Dice's coefficient (Subsection 4.4.1).

Table B.2: Run names used in Chapter 4 (continued).

Run name	Run description
IT-BIN-SIM	Item-based CF using binary vectors from the **IT** matrix and cosine similarity (Subsection 4.4.1).
IT-TF-SIM	Item-based CF using tag frequency vectors from the **IT** matrix and cosine similarity (Subsection 4.4.1).
IT-TFIDF-SIM	Item-based CF using tf·idf-weighted tag frequency vectors from the **IT** matrix and cosine similarity (Subsection 4.4.1).
U-TF-SIM	User-based CF using frequency vectors from the ratings matrix **R** and cosine similarity (Subsection 4.4.2).
U-TFIDF-SIM	User-based CF using tf·idf-weighted frequency vectors from the ratings matrix **R** and cosine similarity (Subsection 4.4.2).
I-TF-SIM	Item-based CF using frequency vectors from the ratings matrix **R** and cosine similarity (Subsection 4.4.2).
I-TFIDF-SIM	Item-based CF using tf·idf-weighted frequency vectors from the ratings matrix **R** and cosine similarity (Subsection 4.4.2).
User-based fusion	User-based CF using fused user-user similarities (Subsection 4.4.3).
Item-based fusion	Item-based CF using fused item-item similarities (Subsection 4.4.3).
U-TSO-SUTTER-SIM	User-based CF using tag frequency vectors from the combined $\mathbf{R}\vert\mathbf{UI}$ matrix and cosine similarity (Subsection 4.6.1).
I-TSO-SUTTER-SIM	Item-based CF using tag frequency vectors from the combined $\mathbf{R}\vert\mathbf{IT}^T$ matrix and cosine similarity (Subsection 4.6.1).
Tag-aware fusion	Weighted combination of U-TSO-SUTTER-SIM and I-TSO-SUTTER-SIM runs (Subsection 4.6.1).
Random walk model	Random walk model according to Clements et al. (2008a) (Subsection 4.6.2).

APPENDIX C

OPTIMAL FUSION WEIGHTS

In this appendix we list the optimal run weights for the ten weighted fusion experiments described in Chapter 6. We list the optimal weights for each of our four data sets separately in Tables C.2 through C.5. Table C.1 lists the IDs we use to refer to the individual runs in those four tables.

Table C.1: Run IDs used in Tables C.2–C.5.

Run ID	Run description
R1	Best user-based run based on usage similarity (from Section 4.3)
R2	Best item-based run based on usage similarity (from Section 4.3)
R3	Best user-based run based on tag overlap similarity (from Subsection 4.4.1)
R4	Best item-based run based on tag overlap similarity (from Subsection 4.4.1)
R5	Best profile-centric run based on metadata (from Subsection 5.2.1)
R6	Best post-centric run based on metadata (from Subsection 5.2.1)
R7	Best user-based run based on metadata (from Subsection 5.2.2)
R8	Best item-based run based on metadata (from Subsection 5.2.2)

Table C.2: Optimal fusion weights for the different fusion experiments on the Bibsonomy bookmarks data set.

Run ID	Method	R1	R2	R3	R4	R5	R6	R7	R8
Fusion A	Weighted CombSUM	0.3	1.0	-	-	-	-	-	-
	Weighted CombMNZ	0.3	1.0	-	-	-	-	-	-
	Weighted CombANZ	0.2	1.0	-	-	-	-	-	-
Fusion B	Weighted CombSUM	-	-	0.4	1.0	-	-	-	-
	Weighted CombMNZ	-	-	0.4	0.8	-	-	-	-
	Weighted CombANZ	-	-	0.4	0.9	-	-	-	-
Fusion C	Weighted CombSUM	0.5	-	-	0.7	-	-	-	-
	Weighted CombMNZ	0.2	-	-	0.8	-	-	-	-
	Weighted CombANZ	0.3	-	-	0.8	-	-	-	-
Fusion D	Weighted CombSUM	-	-	-	-	1.0	0.1	-	-
	Weighted CombMNZ	-	-	-	-	1.0	0.1	-	-
	Weighted CombANZ	-	-	-	-	0.9	1.0	-	-
Fusion E	Weighted CombSUM	-	-	-	-	-	-	0.5	0.7
	Weighted CombMNZ	-	-	-	-	-	-	0.1	0.9
	Weighted CombANZ	-	-	-	-	-	-	0.7	1.0
Fusion F	Weighted CombSUM	-	-	-	-	1.0	-	-	0.4
	Weighted CombMNZ	-	-	-	-	0.5	-	-	0.2
	Weighted CombANZ	-	-	-	-	0.1	-	-	0.9
Fusion G	Weighted CombSUM	-	-	-	0.5	0.7	-	-	-
	Weighted CombMNZ	-	-	-	0.2	0.9	-	-	-
	Weighted CombANZ	-	-	-	1.0	0.2	-	-	-
Fusion H	Weighted CombSUM	0.3	0.9	0.4	0.8	-	-	-	-
	Weighted CombMNZ	0.2	1.0	0.3	1.0	-	-	-	-
	Weighted CombANZ	0.1	1.0	0.2	0.4	-	-	-	-
Fusion I	Weighted CombSUM	-	-	-	-	0.3	0.1	0.5	0.7
	Weighted CombMNZ	-	-	-	-	0.3	0.4	0.2	0.8
	Weighted CombANZ	-	-	-	-	0.3	0.4	0.2	1.0
Fusion J	Weighted CombSUM	0.3	0.8	0.3	0.6	0.9	0.8	0.0	0.8
	Weighted CombMNZ	0.5	0.7	0.4	1.0	0.9	0.4	0.0	0.7
	Weighted CombANZ	0.2	0.6	0.2	0.8	0.0	0.9	0.6	1.0

Table C.3: Optimal fusion weights for the different fusion experiments on the Delicious bookmarks data set.

Run ID	Method	R1	R2	R3	R4	R5	R6	R7	R8
Fusion A	Weighted CombSUM	0.6	1.0	-	-	-	-	-	-
	Weighted CombMNZ	0.9	0.9	-	-	-	-	-	-
	Weighted CombANZ	1.0	0.9	-	-	-	-	-	-
Fusion B	Weighted CombSUM	-	-	0.0	1.0	-	-	-	-
	Weighted CombMNZ	-	-	0.0	1.0	-	-	-	-
	Weighted CombANZ	-	-	0.0	0.9	-	-	-	-
Fusion C	Weighted CombSUM	0.1	-	-	1.0	-	-	-	-
	Weighted CombMNZ	0.0	-	-	0.9	-	-	-	-
	Weighted CombANZ	0.1	-	-	1.0	-	-	-	-
Fusion D	Weighted CombSUM	-	-	-	-	0.0	0.9	-	-
	Weighted CombMNZ	-	-	-	-	0.0	0.7	-	-
	Weighted CombANZ	-	-	-	-	0.0	0.9	-	-
Fusion E	Weighted CombSUM	-	-	-	-	-	-	1.0	0.8
	Weighted CombMNZ	-	-	-	-	-	-	1.0	0.0
	Weighted CombANZ	-	-	-	-	-	-	0.4	1.0
Fusion F	Weighted CombSUM	-	-	-	-	-	0.9	0.6	-
	Weighted CombMNZ	-	-	-	-	-	0.7	0.2	-
	Weighted CombANZ	-	-	-	-	-	1.0	0.5	-
Fusion G	Weighted CombSUM	-	-	-	0.9	-	-	0.0	-
	Weighted CombMNZ	-	-	-	0.9	-	-	0.0	-
	Weighted CombANZ	-	-	-	1.0	-	-	0.1	-
Fusion H	Weighted CombSUM	0.2	0.3	0.1	0.6	-	-	-	-
	Weighted CombMNZ	0.0	0.3	0.0	0.9	-	-	-	-
	Weighted CombANZ	0.4	0.3	0.1	1.0	-	-	-	-
Fusion I	Weighted CombSUM	-	-	-	-	0.0	0.6	0.4	0.9
	Weighted CombMNZ	-	-	-	-	0.0	0.7	0.5	0.9
	Weighted CombANZ	-	-	-	-	0.2	0.9	0.2	0.9
Fusion J	Weighted CombSUM	0.2	0.5	0.1	0.7	0.1	0.5	0.0	0.2
	Weighted CombMNZ	0.7	0.9	0.2	1.0	0.2	0.9	0.1	0.9
	Weighted CombANZ	0.8	0.8	0.2	0.5	0.0	1.0	0.4	0.9

Table C.4: Optimal fusion weights for the different fusion experiments on the Bibsonomy articles data set.

Run ID	Method	R1	R2	R3	R4	R5	R6	R7	R8
Fusion A	Weighted CombSUM	0.9	0.2	-	-	-	-	-	-
	Weighted CombMNZ	0.9	0.2	-	-	-	-	-	-
	Weighted CombANZ	0.8	0.5	-	-	-	-	-	-
Fusion B	Weighted CombSUM	-	-	0.1	0.7	-	-	-	-
	Weighted CombMNZ	-	-	0.0	1.0	-	-	-	-
	Weighted CombANZ	-	-	0.1	0.9	-	-	-	-
Fusion C	Weighted CombSUM	1.0	-	-	0.7	-	-	-	-
	Weighted CombMNZ	1.0	-	-	0.4	-	-	-	-
	Weighted CombANZ	0.3	-	-	0.8	-	-	-	-
Fusion D	Weighted CombSUM	-	-	-	-	0.9	0.3	-	-
	Weighted CombMNZ	-	-	-	-	0.9	0.3	-	-
	Weighted CombANZ	-	-	-	-	0.1	1.0	-	-
Fusion E	Weighted CombSUM	-	-	-	-	-	-	0.1	0.9
	Weighted CombMNZ	-	-	-	-	-	-	0.0	0.9
	Weighted CombANZ	-	-	-	-	-	-	0.1	0.7
Fusion F	Weighted CombSUM	-	-	-	-	0.8	-	-	0.5
	Weighted CombMNZ	-	-	-	-	0.8	-	-	0.5
	Weighted CombANZ	-	-	-	-	0.1	-	-	1.0
Fusion G	Weighted CombSUM	-	-	-	0.7	-	-	-	1.0
	Weighted CombMNZ	-	-	-	0.0	-	-	-	0.9
	Weighted CombANZ	-	-	-	1.0	-	-	-	0.7
Fusion H	Weighted CombSUM	1.0	0.2	0.1	0.7	-	-	-	-
	Weighted CombMNZ	0.8	0.6	0.0	0.2	-	-	-	-
	Weighted CombANZ	0.4	0.4	0.1	0.7	-	-	-	-
Fusion I	Weighted CombSUM	-	-	-	-	0.8	0.2	0.1	0.3
	Weighted CombMNZ	-	-	-	-	0.7	0.6	0.1	1.0
	Weighted CombANZ	-	-	-	-	0.1	0.5	0.1	0.8
Fusion J	Weighted CombSUM	1.0	0.1	0.4	0.4	1.0	0.4	0.0	0.9
	Weighted CombMNZ	0.9	0.3	1.0	0.9	0.5	0.0	0.2	1.0
	Weighted CombANZ	0.6	0.4	0.6	0.8	0.1	0.9	0.2	1.0

Table C.5: Optimal fusion weights for the different fusion experiments on the CiteULike bookmarks data set.

Run ID	Method	R1	R2	R3	R4	R5	R6	R7	R8
Fusion A	Weighted CombSUM	0.2	0.8	-	-	-	-	-	-
	Weighted CombMNZ	0.1	0.6	-	-	-	-	-	-
	Weighted CombANZ	0.1	1.0	-	-	-	-	-	-
Fusion B	Weighted CombSUM	-	-	0.2	0.8	-	-	-	-
	Weighted CombMNZ	-	-	0.1	0.7	-	-	-	-
	Weighted CombANZ	-	-	0.1	0.6	-	-	-	-
Fusion C	Weighted CombSUM	-	0.7	-	0.4	-	-	-	-
	Weighted CombMNZ	-	1.0	-	0.4	-	-	-	-
	Weighted CombANZ	-	0.7	-	0.9	-	-	-	-
Fusion D	Weighted CombSUM	-	-	-	-	1.0	0.2	-	-
	Weighted CombMNZ	-	-	-	-	1.0	0.2	-	-
	Weighted CombANZ	-	-	-	-	0.0	1.0	-	-
Fusion E	Weighted CombSUM	-	-	-	-	-	-	0.1	0.9
	Weighted CombMNZ	-	-	-	-	-	-	0.1	1.0
	Weighted CombANZ	-	-	-	-	-	-	0.1	0.8
Fusion F	Weighted CombSUM	-	-	-	-	0.7	-	-	0.5
	Weighted CombMNZ	-	-	-	-	1.0	-	-	0.4
	Weighted CombANZ	-	-	-	-	0.1	-	-	0.9
Fusion G	Weighted CombSUM	-	1.0	-	-	0.7	-	-	-
	Weighted CombMNZ	-	1.0	-	-	0.7	-	-	-
	Weighted CombANZ	-	1.0	-	-	0.1	-	-	-
Fusion H	Weighted CombSUM	0.3	0.8	0.0	1.0	-	-	-	-
	Weighted CombMNZ	0.1	0.8	0.0	0.5	-	-	-	-
	Weighted CombANZ	0.1	0.9	0.0	0.5	-	-	-	-
Fusion I	Weighted CombSUM	-	-	-	-	1.0	0.1	0.1	0.3
	Weighted CombMNZ	-	-	-	-	0.8	0.2	0.1	0.1
	Weighted CombANZ	-	-	-	-	0.0	0.9	0.1	0.9
Fusion J	Weighted CombSUM	0.4	1.0	0.3	0.7	1.0	0.5	0.0	0.7
	Weighted CombMNZ	0.4	1.0	0.1	0.2	0.2	0.8	0.0	0.1
	Weighted CombANZ	0.3	1.0	0.6	1.0	0.1	1.0	0.4	1.0

APPENDIX D

DUPLICATE ANNOTATION IN CITEULIKE

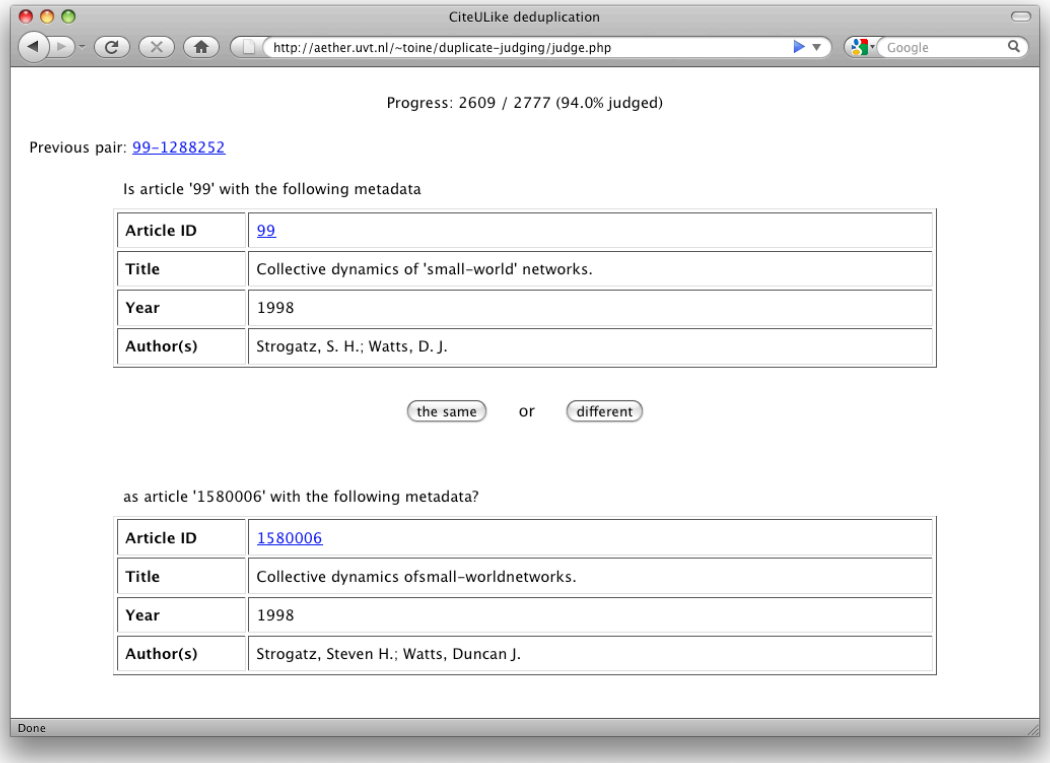


Figure D.1: Screenshot of the interface for judging duplicate CiteULike pairs. The example pair in the screenshot is a typical example of a duplicate pair. For each article in the pair, we display the hyperlinked article ID pointing to the CiteULike article page, the article title, the publication year, and the author(s).

After obtaining a training set of 2,777 pairs as described in Subsection 8.3.1, we needed to determine which of these pairs were duplicates and which were different items. Figure D.1 shows the simple interface we created for duplicate annotation. The seed item is shown at the top with its item ID, title, year, and author metadata and the same metadata fields are shown at the bottom of the screen for the matched item. The annotator can then judge the

articles to be ‘the same’ or ‘different’. The article IDs are linked to their CiteULike article pages for easy reference.

List of Figures

List of Tables

List of Abbreviations

AUC	**A**rea **U**nder the **C**urve
CAPTCHA	**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part
CBR	**C**ase-**B**ased **R**easoning
CF	**C**ollaborative **F**iltering
DOI	**D**igital **O**bject **I**dentifier
GBCF	**G**raph-**B**ased **C**ollaborative **F**iltering
HRI	**H**uman-**R**ecommender **I**nteraction
HTML	**H**yper-**T**ext **M**arkup **L**anguage
IB	**I**tem-**B**ased filtering
IF	**I**nformation **F**iltering
IMA	**I**nformation **M**anagement **A**ssistant
IR	**I**nformation **R**etrieval
ISBN	**I**nternational **S**tandard **B**ook **N**umber
ISSN	**I**nternational **S**tandard **S**erial **N**umber
IT	**I**nformation **T**echnology
KL	**K**ullback-**L**eibler divergence
MAP	**M**ean **A**verage **P**recision
MRR	**M**ean **R**eciprocal **R**ank
PDF	**P**ortable **D**ocument **F**ormat
PLSA	**P**robabilistic **L**atent **S**emantic **A**nalysis
ROC	**R**eceiver-**O**perator **C**haracteristic
SGML	**S**tandard **G**eneralized **M**arkup **L**anguage
SSI	**S**et of **S**imilar **I**tems
SSU	**S**et of **S**imilar **U**sers
SVM	**S**upport **V**ector **M**achine
TBCF	**T**ag-**B**ased **C**ollaborative **F**iltering
TIBCF	**T**agging **I**ntensity-**B**ased **C**ollaborative **F**iltering
TOBCF	**T**ag **O**verlap-**B**ased **C**ollaborative **F**iltering
TREC	**T**ext **RE**trieval **C**onference
UB	**U**ser-**B**ased filtering
UCOV	**U**ser **COV**erage
URL	**U**niform **R**esource **L**ocator

SUMMARY

Under the label of *Web 2.0*, the past decade has seen a genuine and fundamental change in the way people interact with and through the World Wide Web. One of its prominent features is a shift in information access and creation from local and solitary, to global and collaborative. *Social bookmarking websites* such as Delicious and CiteULike are a clear case in point of this shift: instead of keeping a local copy of pointers to favorite URLs or scientific articles, users can instead store and access their bookmarks online through a Web interface. The underlying system then makes all stored information shareable among users. In addition to this functionality, most social bookmarking services also offer the user the opportunity to describe the content they added to their personal profile with keywords. These keywords, commonly referred to as *tags*, are an addition to e.g. the title and summary metadata commonly used to annotate content, and improve the access and retrievability of a user's own bookmarked Web pages. These tags are then made available to all users, many of whom have annotated many of the same Web pages with possibly overlapping tags. This *social tagging* results in a rich network of users, bookmarks, and tags, commonly referred to as a *folksonomy*.

As social bookmarking systems become more popular, they need effective access methods to help users locate all the interesting content present in those systems. One such type of technology are *recommender systems*, which are a form of personalized information filtering technology used to identify sets of items that are likely to be of interest to a certain user, using a variety of information sources. In this thesis, we investigate how recommender systems can be applied to the domain of social bookmarking to recommend interesting and relevant Web pages and scientific articles to the user, based on two main information sources about the user and the items: *usage data* and *metadata*. The former represent the past selections and transactions of all users, while the latter describe the resources on a social bookmarking website with e.g. titles, descriptions, authorship, tags, and temporal and publication-related metadata. These two characteristic information sources of social bookmarking lead us to formulate the following problem statement for this thesis:

> *How can we utilize the characteristics of social bookmarking websites to produce the best possible item recommendations for users?*

We start in Chapter 2 by providing the reader with an overview of the general related work on recommender systems. We give a historical overview, describe the most important approaches and specific discuss related approaches to recommending Web pages and scientific articles. We discuss the history of social tagging and its applications, and we give an overview of the history and structure of social bookmarking websites.

Chapter 3 describes our research methodology. We define the recommendation task we are trying to solve, and introduce our four data sets and their collection process. We also discuss our experimental setup, from data pre-processing and filtering to our choice of evaluation metrics.

In Chapter 4 we investigate the first of two important characteristics of social bookmarking systems: the presence of the folksonomy. We focus on using the tags present in the folksonomy of social bookmarking systems, which describe the content of an item and can therefore be used in determining the similarity between two objects. We extend a standard nearest-neighbor collaborative filtering algorithm with different tag similarity metrics. We find that, because of reduced sparsity, the performance of item-based filtering can be improved by using the item similarities based on the overlap in the tags assigned to those items. User-based filtering is not helped by using tag overlap similarities. We find that it is easier to recommend scientific articles than bookmarks. We also examine merging different types of similarity with inconclusive results, and compared our algorithms with two state-of-the-art approaches. We conclude that tags can be used successfully to improve performance.

In Chapter 5, we investigate the usefulness of item metadata in recommendation process, the other characteristic of social bookmarking websites, and examine how we can use this metadata to improve recommendation performance. We propose four different algorithms, divided into two classes: two content-based filtering approaches and two hybrid approaches. In content-based filtering, a profile-centric approach, where all of the metadata assigned by a user is matched against metadata representations of the items, works better than matching posts with each other because of sparseness issues. We also compare two hybrid CF approaches that use the metadata representations to calculate the user and item similarities. Here, we find that item-based filtering with the metadata-derived similarities works best. What the best overall metadata-based algorithm is, is dependent on the data set. We also find that the quality of the metadata is as important to performance as the sparsity of the metadata field. Depending on the data set, metadata-based recommendation works as well as folksonomic recommendation or better.

Chapter 6 describes our experiments with combining the different recommendation algorithms from the previous chapters. We compared weighted and unweighted methods of combining the predictions of different algorithms and find that weighted fusion is superior to unweighted fusion. The best results are achieved by fusing the results of recommendation algorithms and representations that touch upon different aspects of the item recommendation process.

In addition to the recommendation experiments described in Chapters 4 through 6, we also examine two specific growing pains that accompany the increasing popularity of social bookmarking websites. Chapter 7 examines the problem of *spam* for two of our collections, and find they contain large amounts of spam, ranging from 30% to 93% of all users marked

as spammers. We show that it is possible to train a classifier to automatically detect spam users in a social bookmarking system, by comparing all of the metadata they have added together to the metadata added by genuine users and by other spammers. We also examine the influence of spam on recommendation performance and find that spam has a negative effect on recommendation. The influence of spam depends on the recommendation algorithm, but all result lists are unacceptably polluted with spam items, proving the necessity of adequate spam detection techniques.

Chapter 8 examines the problem of *duplicate content*, i.e. when users accidentally and carelessly add Web pages or scientific articles that are already present in the system. We examine one of our data sets to quantify the problem of duplicate content. We construct a training set and train a duplicate identification classifier which found a small percentage of duplicates. We find that these duplicate items follow a Zipfian distribution with a long tail, just as regular items do, which means that certain duplicates can be quite widespread. Finally, we examine the influence of duplicates on recommendation performance by creating a deduplicated version of our data set. We two different recommendation algorithms, but do not find any clear effect of deduplication on recommendation.

SAMENVATTING

Onder de noemer *Web 2.0* is er in de afgelopen jaren een fundamentele verandering opgetreden in de manier waarop mensen met elkaar communiceren en interageren op en via het World Wide Web. Een van de meest prominente eigenschappen hiervan is een verschuiving in de toegang tot informatie van lokaal en individueel tot globaal en collaboratief. Zogenaamde *social bookmarking* diensten als Delicious en CiteULikezijn hier een duidelijk voorbeeld van. Social bookmarking websites staan hun gebruikers toe om hun favoriete webpagina's or wetenschappelijke artikelen online op te slaan en te beheren middels een web-interface, in plaats van deze 'bookmarks' lokaal in een browser op te slaan. Het onderliggende systeem maakt dan alle opgeslagen informatie toegankelijk voor alle gebruikers. Naast deze functionaliteit bieden de meeste social bookmarking diensten de gebruiker ook de mogelijk om de objecten te beschrijven die ze aan hun profiel hebben toegevoegd te beschrijven met steekwoorden. Deze steekwoorden worden ook wel *tags* genoemd en vormen een aanvulling bovenop de gebruikelijke metadata zoals titel en samenvatting. Tags kunnen gebruikt worden om de bookmarks van een gebruiker te categoriseren en verhogen de toegang tot en de hervindbaarheid van de bookmarks. De tags die een gebruiker toekent, worden beschikbaar gemaakt voor alle gebruikers, waarvan velen dezelfde webpagina's toe hebben gevoegd met mogelijkerwijs overlappende tags. Dit zogenaamde *social tagging* fenomeen leidt tot het onstaan van een rijk netwerk van gebruikers, bookmarks en tags, ook wel een *folksonomie* genoemd.

De groei in populariteit van social bookmarking systemen brengt een behoefte met zich mee aan effectieve methoden en technieken waarmee gebruikers nieuwe en interessante inhoud kunnen vinden in dit soort systemen. Een voorbeeld van dit soort technologie wordt gevormd door *recommendersystemen*, die het filteren van informatie personaliseren voor de gebruiker en op die manier die objecten kunnen identificeren die voor de gebruiker interessant zouden kunnen zijn. Dit soort suggesties kunnen gebaseerd worden op een verscheidenheid aan informatiebronnen. In dit proefschrift wordt onderzocht hoe recommendersystemen toegepast kunnen worden op het domein van social bookmarking om interessante webpagina's en wetenschappelijke artikelen aan te raden aan gebruikers. Deze suggesties worden gegenereerd op basis van twee informatiebronnen over de gebruikers en de objecten: *gebruiksinformatie* en *metadata*. Gebruiksinformatie biedt een overzicht van de transacties en selecties die gebruikers in het verleden hebben uitgevoerd, d.w.z. welke objecten hebben ze toegevoegd aan hun profiel en met welke tags? Metadata beschrijft

de objecten op een social bookmarking website met bijvoorbeeld informatie over titels, samenvattingen of auteursschap en met tags en tijds- en publicatiegerelateerde informatie. Deze twee voor social bookmarking karakteristieke informatiebronnen brengen ons ertoe de volgende probleemstelling te formuleren:

> *Hoe kunnen we het beste de karakteristieken van social bookmarking websites benutten om de beste mogelijke suggesties te genereren voor gebruikers van dit soort systemen?*

We beginnen ons onderzoek in Hoofdstuk 2, waar we de lezer een historisch overzicht bieden van recommendersystemen. We beschrijven de belangrijkste aanpakken en gaan dieper in op gerelateerde werk aan systemen die webpagina's of artikelen kunnen voorstellen aan gebruikers. We beschrijven ook de geschiedenis van social tagging en haar toepassingen en we geven een overzicht van de historie en de structuur van social bookmarking websites.

In Hoofdstuk 3 wordt de methodologie beschreven die in onze experimenten gevolgd wordt. We preciseren de taak die we willen vervullen en introduceren de vier datasets waarop onze experimenten zijn gebaseerd. We bespreken tevens onze experimentele opzet, van het voorbewerken van onze data en het filteren ervan, tot onze keuze voor de juiste evaluatiemetrieken.

Hoofdstuk 4 onderzoekt de eerste van de twee eerdergenoemde belangrijke karakteristieken van social bookmarking systemen: de aanwezigheid van een folksonomie. Voor het bepalen van de overeenkomst tussen objecten of gebruikers richten we ons op de tags waarmee gebruikers de inhoud van objecten beschrijven. We breiden met succes een standaard geheugengebaseerd recommendation algoritme uit met verschillende 'similarity' metrieken gebaseerd op tag-overlap. Een vergelijking met bestaande algoritmes toont aan dat onze aanpak competitief is hiermee.

In Hoofstuk 5 onderzoeken we het nut van het gebruiken van object metadata in het aanraden van webpagina's en wetenschappelijke artikelen. We presenteren vier verschillende algoritmes die deze informatiebron gebruiken om suggesties te genereren. Welk algoritme het beste werkt is afhankelijk van de data set en domein, maar uit onze experimenten blijkt dat de kwaliteit van de toegewezen metadata net zo belangrijk is voor de prestaties van de algoritmes als de schaarste van de beschikbare metadatavelden. Vergeleken met recommendation op basis van de folksonomie werken bepaalde algoritmes gebaseerd op metadata net zo goed of beter.

In Hoofdstuk 6 beschrijven we experimenten waarin we de verschillende algoritmes uit de voorgaande hoofdstukken combineren. We vergelijken gewogen en ongewogen methoden om de voorspellingen van de verschillende algoritmes te combineren; gewogen combinatie blijkt de meest veelbelovende fusiemethode te zijn. De beste resultaten worden behaald door het fuseren van verschillende algoritmes en objectrepresentaties die zoveel mogelijk verschillende aspecten van het recommendation-proces benadrukkken.

Naast de experimenten in Hoofdstuk 4 tot en met 6 onderzoeken we tevens twee specifieke problemen waar snelgroeiende social bookmarking websites mee te maken kunnen krijgen.

In hoofdstuk 7 onderzoeken we of *spam* een groot probleem is voor twee van onze datasets. Met een percentage tussen de 30% en 93% aan spammers blijkt dit een veelvoorkomend probleem te zijn. We tonen aan dat het mogelijk is om een classificatiealgoritme te trainen dat automatisch spammers kan detecteren door de door hen toegewezen metadata te vergelijken met die van echte gebruikers en andere spammers. We constateren verder dat spam een negatieve invloed heeft op het aanraden van interessante webpagina's voor gebruikers, Alhoewel de invloed afhankelijk is van het gekozen recommendation-algoritme, worden de lijsten met gesuggereerde webpagina's onacceptabel vervuild met spam. Dit toont de noodzaak aan van effective technieken voor het detecteren van spam.

Hoofdstuk 8 onderzoekt tenslotte het probleem van *duplicate content*, d.w.z. objecten die per ongeluk dubbel toegevoegd worden aan een social bookmarking systeem door gebruikers. We onderzoeken een van onze datasets om een beeld van dit probleem te krijgen en merken dat, alhoewel de meeste duplicaten slechts enkele keren voorkomen, bepaalde duplicaten tamelijk populair kunnen worden. Daarnaast creëren we een trainingsset om een automatisch classificatiealgoritme te trainen dat dubbele objecten kan leren herkennen. Tenslotte onderzoeken we de invloed van duplicate content op de recommendation-algoritmes, maar vinden hier geen sterk effect op de uiteindelijke prestaties.

Curriculum Vitae

Toine Bogers was born in Roosendaal, the Netherlands, on the 21st of September, 1979. After graduating high school at Gertrudiscollege in Roosendaal in 1997, he started studying Information Management at Tilburg University and obtained his M.A. degree in 2001. His M.A. thesis was about a comparison of major Web service frameworks. In 2002, he started his studies in Computational Linguistics & AI at Tilburg University; two years later, he obtained a second M.A. degree. His second M.A. thesis was about the optimization of a named entity recognizer for Dutch.

In December 2004, he joined the Induction of Linguistic Knowledge group at Tilburg University as a Ph.D. student under the supervision of Antal van den Bosch. Funded by SenterNovem and the Dutch Ministry of Economic Affairs as part of the IOP-MMI À Propos project, the goal of this project was to develop an adaptive, personalized, just-in-time knowledge management environment for members of professional workgroups, which supports them in writing documents and test its usability in actual offices. The project was carried out in collaboration with the University of Nijmegen and Intelli-Gent. The focus of Toine's work was on the personalization and recommendation of relevant document by the information agent.

In addition, Toine worked on expert search with researchers of the ILPS group at the University of Amsterdam, which culminated in the development and adoption of a university-wide expert search engine for Tilburg University. Another project was developing a system for recommendation related news articles for the website of the Dutch newspaper Trouw. In 2008, Toine spent two and a half months working as a guest researcher at the Royal School of Library and Information Science in Copenhagen, Denmark.

As of October 2009, Toine has been working at the Royal School of Library and Information Science in Copenhagen, Denmark. His research interests include recommender systems, enterprise search, information retrieval, intelligent multimedia information systems (content analysis, retrieval, and personalization).

PUBLICATIONS

The investigations performed during my Ph.D. research resulted in the following publications.

- K. Hofmann, K. Balog, T. Bogers, and M. de Rijke. Contextual Factors for Similar Expert Finding. To appear in *Journal of the American Society for Information Science*, 2010.
- T. Bogers and A. van den Bosch. Collaborative and Content-based Filtering for Item Recommendation on Social Bookmarking Websites. To appear in *Proceedings of the ACM RecSys '09 workshop on Recommender Systems and the Social Web*, 2009.
- R. Liebregts and T. Bogers. Design and Implementation of a University-wide Expert Search Engine. In M. Boughanem et al., editors, *Proceedings of the 31st European Conference on Information Retrieval (ECIR 2009), volume 5478 of Lecture Notes in Computer Science*, pages 587–594, 2009.
- T. Bogers and A. van den Bosch. Using Language Modeling for Spam Detection in Social Reference Manager Websites. In R. Aly et al., editors, *Proceedings of the 9th Belgian-Dutch Information Retrieval Workshop (DIR 2009)*, pages 87–94, 2009.
- T. Bogers and A. van den Bosch. Recommending Scientific Articles using CiteULike. In *RecSys '08: Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 287–290. 2008.
- T. Bogers and A. van den Bosch. Using Language Models for Spam Detection in Social Bookmarking. In *Proceedings of 2008 ECML/ PKDD Discovery Challenge Workshop*, pages 1–12, 2008.
- A. van den Bosch and T. Bogers. Efficient Context-Sensitive Word Completion for Mobile Devices. In *MobileHCI 2008: Proceedings of the 10th International Conference on Human-Computer Interaction with Mobile Devices and Services*, IOP-MMI special track, pages 465–470, 2008.
- K. Hofmann, K. Balog, T. Bogers, and M. de Rijke. Integrating Contextual Factors into Topic-centric Retrieval Models for Finding Similar Experts. In *Proceedings of ACM SIGIR 2008 Workshop on Future Challenges in Expert Retrieval*, pages 29–36, 2008.
- M.C. Puerta Melguizo, O. Muñoz Ramos, L. Boves, T. Bogers, and A. van den Bosch. A Personalized Recommender System for Writing in the Internet Age. In *Proceedings of the LREC 2008 workshop on Natural Language Processing Resources, Algorithms, and Tools for Authoring Aids*, 2008.

- T. Bogers, K. Kox, and A. van den Bosch. Using Citation Analysis for Finding Experts in Workgroups. In E. Hoenkamp et al., editors, *Proceedings of the 8th Belgian-Dutch Information Retrieval Workshop (DIR 2008)*, pages 21–28, 2008.
- T. Bogers and A. van den Bosch. Comparing and Evaluating Information Retrieval Algorithms for News Recommendation. In *RecSys '07: Proceedings of the 2007 ACM Conference on Recommender Systems*, pages 141–144, 2007.
- K. Balog, T. Bogers, L. Azzopardi, M. de Rijke, and A. van den Bosch. Broad Expertise Retrieval in Sparse Data Environments. In C. Clarke et al., editors, *SIGIR '07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 551–558, 2007.
- T. Bogers, W. Thoonen and A. van den Bosch. Expertise Classification: Collaborative Classification vs. Automatic Extraction. In J.T. Tennis et al., editors, *Proceedings of the 17th ASIS&T SIG/CR workshop on Social Classification*, 2006.
- T. Bogers. À Propos: Pro-Active Personalization for Professional Document Writing (long abstract). In I. Ruthven et al., editors, *Proceedings of the First IIiX Symposium on Information Interaction in Context (IIiX 2006)*, page 303, 2006.
- S. Canisius, T. Bogers, A. van den Bosch, J. Geertzen, and E. Tjong Kim Sang. Dependency Parsing by Inference over High-recall Dependency Predictions. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, 2006.
- T. Bogers and A. van den Bosch. Authoritative Re-ranking of Search Results. In M. Lalmas et al., editors, *Proceedings of the 28th European Conference on Information Retrieval (ECIR 2006)*, volume 3936 of Lecture Notes in Computer Science, pages 519–522, 2006.
- T. Bogers and A. van den Bosch. Authoritative Re-ranking in Fusing Authorship-based Subcollection Search Results. In Franciska de Jong and Wessel Kraaij, editors, *Proceedings of the Sixth Belgian- Dutch Information Retrieval Workshop (DIR 2006)*, pages 49–55, 2006.
- E. Tjong Kim Sang, S. Canisius, A. van den Bosch, and T. Bogers. Applying Spelling Error Correction Techniques for Improving Semantic Role Labelling. In *Proceedings of the Ninth Conference on Natural Language Learning (CoNLL-2005)*, 2005.

SIKS Dissertation Series

1998

1 Johan van den Akker (CWI[1]) *DEGAS - An Active, Temporal Database of Autonomous Objects*

2 Floris Wiesman (UM) *Information Retrieval by Graphically Browsing Meta-Information*

3 Ans Steuten (TUD) *A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective*

4 Dennis Breuker (UM) *Memory versus Search in Games*

5 Eduard W. Oskamp (RUL) *Computerondersteuning bij Straftoemeting*

1999

1 Mark Sloof (VU) *Physiology of Quality Change Modelling; Automated Modelling of Quality Change of Agricultural Products*

2 Rob Potharst (EUR) *Classification using Decision Trees and Neural Nets*

3 Don Beal (UM) *The Nature of Minimax Search*

4 Jacques Penders (UM) *The Practical Art of Moving Physical Objects*

5 Aldo de Moor (KUB) *Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems*

6 Niek J.E. Wijngaards (VU) *Re-Design of Compositional Systems*

7 David Spelt (UT) *Verification Support for Object Database Design*

8 Jacques H.J. Lenting (UM) *Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation*

2000

1 Frank Niessink (VU) *Perspectives on Improving Software Maintenance*

2 Koen Holtman (TU/e) *Prototyping of CMS Storage Management*

3 Carolien M.T. Metselaar (UvA) *Sociaal-organisatorische Gevolgen van Kennistechnologie; een Procesbenadering en Actorperspectief*

4 Geert de Haan (VU) *ETAG, A Formal Model of Competence Knowledge for User Interface Design*

5 Ruud van der Pol (UM) *Knowledge-Based Query Formulation in Information Retrieval*

6 Rogier van Eijk (UU) *Programming Languages for Agent Communication*

7 Niels Peek (UU) *Decision-Theoretic Planning of Clinical Patient Management*

8 Veerle Coupé (EUR) *Sensitivity Analyis of Decision-Theoretic Networks*

9 Florian Waas (CWI) *Principles of Probabilistic Query Optimization*

10 Niels Nes (CWI) *Image Database Management System Design Considerations, Algorithms and Architecture*

11 Jonas Karlsson (CWI) *Scalable Distributed Data Structures for Database Management*

2001

1 Silja Renooij (UU) *Qualitative Approaches to Quantifying Probabilistic Networks*

2 Koen Hindriks (UU) *Agent Programming Languages: Programming with Mental Models*

[1] Abbreviations: SIKS - Dutch Research School for Information and Knowledge Systems; CWI - Centrum voor Wiskunde en Informatica, Amsterdam; EUR - Erasmus Universiteit, Rotterdam; KUB - Katholieke Universiteit Brabant, Tilburg; KUN - Katholieke Universiteit Nijmegen; OU - Open Universiteit; RUL - Rijksuniversiteit Leiden; RUN - Radboud Universiteit Nijmegen; TUD - Technische Universiteit Delft; TU/e - Technische Universiteit Eindhoven; UL - Universiteit Leiden; UM - Universiteit Maastricht; UT - Universiteit Twente, Enschede; UU - Universiteit Utrecht; UvA - Universiteit van Amsterdam; UvT - Universiteit van Tilburg; VU - Vrije Universiteit, Amsterdam.

3 Maarten van Someren (UvA) *Learning as Problem Solving*

4 Evgueni Smirnov (UM) *Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets*

5 Jacco van Ossenbruggen (VU) *Processing Structured Hypermedia: A Matter of Style*

6 Martijn van Welie (VU) *Task-Based User Interface Design*

7 Bastiaan Schonhage (VU) *Diva: Architectural Perspectives on Information Visualization*

8 Pascal van Eck (VU) *A Compositional Semantic Structure for Multi-Agent Systems Dynamics*

9 Pieter Jan 't Hoen (RUL) *Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes*

10 Maarten Sierhuis (UvA) *Modeling and Simulating Work Practice BRAHMS: a Multiagent Modeling and Simulation Language for Work Practice Analysis and Design*

11 Tom M. van Engers (VU) *Knowledge Management: The Role of Mental Models in Business Systems Design*

2002

1 Nico Lassing (VU) *Architecture-Level Modifiability Analysis*

2 Roelof van Zwol (UT) *Modelling and Searching Web-based Document Collections*

3 Henk Ernst Blok (UT) *Database Optimization Aspects for Information Retrieval*

4 Juan Roberto Castelo Valdueza (UU) *The Discrete Acyclic Digraph Markov Model in Data Mining*

5 Radu Serban (VU) *The Private Cyberspace Modeling Electronic Environments Inhabited by Privacy-Concerned Agents*

6 Laurens Mommers (UL) *Applied Legal Epistemology; Building a Knowledge-based Ontology of the Legal Domain*

7 Peter Boncz (CWI) *Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications*

8 Jaap Gordijn (VU) *Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas*

9 Willem-Jan van den Heuvel (KUB) *Integrating Modern Business Applications with Objectified Legacy Systems*

10 Brian Sheppard (UM) *Towards Perfect Play of Scrabble*

11 Wouter C.A. Wijngaards (VU) *Agent Based Modelling of Dynamics: Biological and Organisational Applications*

12 Albrecht Schmidt (UvA) *Processing XML in Database Systems*

13 Hongjing Wu (TU/e) *A Reference Architecture for Adaptive Hypermedia Applications*

14 Wieke de Vries (UU) *Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems*

15 Rik Eshuis (UT) *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*

16 Pieter van Langen (VU) *The Anatomy of Design: Foundations, Models and Applications*

17 Stefan Manegold (UvA) *Understanding, Modeling, and Improving Main-Memory Database Performance*

2003

1 Heiner Stuckenschmidt (VU) *Ontology-Based Information Sharing in Weakly Structured Environments*

2 Jan Broersen (VU) *Modal Action Logics for Reasoning About Reactive Systems*

3 Martijn Schuemie (TUD) *Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy*

4 Milan Petkovic (UT) *Content-Based Video Retrieval Supported by Database Technology*

5 Jos Lehmann (UvA) *Causation in Artificial Intelligence and Law – A Modelling Approach*

6 Boris van Schooten (UT) *Development and Specification of Virtual Environments*

7 Machiel Jansen (UvA) *Formal Explorations of Knowledge Intensive Tasks*

8 Yong-Ping Ran (UM) *Repair-Based Scheduling*

9 Rens Kortmann (UM) *The Resolution of Visually Guided Behaviour*

10 Andreas Lincke (UT) *Electronic Business Negotiation: Some Experimental Studies on the Interaction between Medium, Innovation Context and Cult*

11 Simon Keizer (UT) *Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks*

12 Roeland Ordelman (UT) *Dutch Speech Recognition in Multimedia Information Retrieval*

13 Jeroen Donkers (UM) *Nosce Hostem – Searching with Opponent Models*

14 Stijn Hoppenbrouwers (KUN) *Freezing Language: Conceptualisation Processes across ICT-Supported Organisations*

15 Mathijs de Weerdt (TUD) *Plan Merging in Multi-Agent Systems*

16 Menzo Windhouwer (CWI) *Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouse*

17 David Jansen (UT) *Extensions of Statecharts with Probability, Time, and Stochastic Timing*

18 Levente Kocsis (UM) *Learning Search Decisions*

2004

1 Virginia Dignum (UU) *A Model for Organizational Interaction: Based on Agents, Founded in Logic*

2 Lai Xu (UvT) *Monitoring Multi-party Contracts for E-business*

3 Perry Groot (VU) *A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving*

4 Chris van Aart (UvA) *Organizational Principles for Multi-Agent Architectures*

5 Viara Popova (EUR) *Knowledge Discovery and Monotonicity*

6 Bart-Jan Hommes (TUD) *The Evaluation of Business Process Modeling Techniques*

7 Elise Boltjes (UM) *Voorbeeld$_{IG}$ Onderwijs; Voorbeeldgestuurd Onderwijs, een Opstap naar Abstract Denken, vooral voor Meisjes*

8 Joop Verbeek (UM) *Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale Politiële Gegevensuitwisseling en Digitale Expertise*

9 Martin Caminada (VU) *For the Sake of the Argument; Explorations into Argument-based Reasoning*

10 Suzanne Kabel (UvA) *Knowledge-rich Indexing of Learning-objects*

11 Michel Klein (VU) *Change Management for Distributed Ontologies*

12 The Duy Bui (UT) *Creating Emotions and Facial Expressions for Embodied Agents*

13 Wojciech Jamroga (UT) *Using Multiple Models of Reality: On Agents who Know how to Play*

14 Paul Harrenstein (UU) *Logic in Conflict. Logical Explorations in Strategic Equilibrium*

15 Arno Knobbe (UU) *Multi-Relational Data Mining*

16 Federico Divina (VU) *Hybrid Genetic Relational Search for Inductive Learning*

17 Mark Winands (UM) *Informed Search in Complex Games*

18 Vania Bessa Machado (UvA) *Supporting the Construction of Qualitative Knowledge Models*

19 Thijs Westerveld (UT) *Using generative probabilistic models for multimedia retrieval*

20 Madelon Evers (Nyenrode) *Learning from Design: facilitating multidisciplinary design teams*

2005

1 Floor Verdenius (UvA) *Methodological Aspects of Designing Induction-Based Applications*

2 Erik van der Werf (UM) *AI techniques for the game of Go*

3 Franc Grootjen (RUN) *A Pragmatic Approach to the Conceptualisation of Language*

4 Nirvana Meratnia (UT) *Towards Database Support for Moving Object data*

5 Gabriel Infante-Lopez (UvA) *Two-Level Probabilistic Grammars for Natural Language Parsing*

6 Pieter Spronck (UM) *Adaptive Game AI*

7 Flavius Frasincar (TU/e) *Hypermedia Presentation Generation for Semantic Web Information Systems*

8 Richard Vdovjak (TU/e) *A Model-driven Approach for Building Distributed Ontology-based Web Applications*

9 Jeen Broekstra (VU) *Storage, Querying and Inferencing for Semantic Web Languages*

10 Anders Bouwer (UvA) *Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments*

11 Elth Ogston (VU) *Agent Based Matchmaking and Clustering - A Decentralized Approach to Search*

12 Csaba Boer (EUR) *Distributed Simulation in Industry*

13 Fred Hamburg (UL) *Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen*

14 Borys Omelayenko (VU) *Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics*

15 Tibor Bosse (VU) *Analysis of the Dynamics of Cognitive Processes*

16 Joris Graaumans (UU) *Usability of XML Query Languages*

17 Boris Shishkov (TUD) *Software Specification Based on Re-usable Business Components*

18 Danielle Sent (UU) *Test-selection strategies for probabilistic networks*

19 Michel van Dartel (UM) *Situated Representation*

20 Cristina Coteanu (UL) *Cyber Consumer Law, State of the Art and Perspectives*

21 Wijnand Derks (UT) *Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics*

2006

1 Samuil Angelov (TU/e) *Foundations of B2B Electronic Contracting*

2 Cristina Chisalita (VU) *Contextual issues in the design and use of information technology in organizations*

3 Noor Christoph (UvA) *The role of metacognitive skills in learning to solve problems*

4 Marta Sabou (VU) *Building Web Service Ontologies*

5 Cees Pierik (UU) *Validation Techniques for Object-Oriented Proof Outlines*

6 Ziv Baida (VU) *Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling*

7 Marko Smiljanic (UT) *XML schema matching – balancing efficiency and effectiveness by means of clustering*

8 Eelco Herder (UT) *Forward, Back and Home Again - Analyzing User Behavior on the Web*

9 Mohamed Wahdan (UM) *Automatic Formulation of the Auditor's Opinion*

10 Ronny Siebes (VU) *Semantic Routing in Peer-to-Peer Systems*

11 Joeri van Ruth (UT) *Flattening Queries over Nested Data Types*

12 Bert Bongers (VU) *Interactivation - Towards an e-cology of people, our technological environment, and the arts*

13 Henk-Jan Lebbink (UU) *Dialogue and Decision Games for Information Exchanging Agents*

14 Johan Hoorn (VU) *Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change*

15 Rainer Malik (UU) *CONAN: Text Mining in the Biomedical Domain*

16 Carsten Riggelsen (UU) *Approximation Methods for Efficient Learning of Bayesian Networks*

17 Stacey Nagata (UU) *User Assistance for Multitasking with Interruptions on a Mobile Device*

18 Valentin Zhizhkun (UvA) *Graph transformation for Natural Language Processing*

19 Birna van Riemsdijk (UU) *Cognitive Agent Programming: A Semantic Approach*

20 Marina Velikova (UvT) *Monotone models for prediction in data mining*

21 Bas van Gils (RUN) *Aptness on the Web*

22 Paul de Vrieze (RUN) *Fundaments of Adaptive Personalisation*

23 Ion Juvina (UU) *Development of Cognitive Model for Navigating on the Web*

24 Laura Hollink (VU) *Semantic Annotation for Retrieval of Visual Resources*

25 Madalina Drugan (UU) *Conditional log-likelihood MDL and Evolutionary MCMC*

26 Vojkan Mihajlovic (UT) *Score Region Algebra: A Flexible Framework for Structured Information Retrieval*

27 Stefano Bocconi (CWI) *Vox Populi: generating video documentaries from semantically annotated media repositories*

28 Borkur Sigurbjornsson (UvA) *Focused Information Access using XML Element Retrieval*

2007

1 Kees Leune (UvT) *Access Control and Service-Oriented Architectures*

2 Wouter Teepe (RUG) *Reconciling Information Exchange and Confidentiality: A Formal Approach*

3 Peter Mika (VU) *Social Networks and the Semantic Web*

4 Jurriaan van Diggelen (UU) *Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach*

5 Bart Schermer (UL) *Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance*

6 Gilad Mishne (UvA) *Applied Text Analytics for Blogs*

7 Natasa Jovanovic' (UT) *To Whom It May Concern - Addressee Identification in Face-to-Face Meetings*

8 Mark Hoogendoorn (VU) *Modeling of Change in Multi-Agent Organizations*

9 David Mobach (VU) *Agent-Based Mediated Service Negotiation*

10 Huib Aldewereld (UU) *Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols*

11 Natalia Stash (TU/e) *Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System*

12 Marcel van Gerven (RUN) *Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty*

13 Rutger Rienks (UT) *Meetings in Smart Environments; Implications of Progressing Technology*

14 Niek Bergboer (UM) *Context-Based Image Analysis*

15 Joyca Lacroix (UM) *NIM: a Situated Computational Memory Model*

16 Davide Grossi (UU) *Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems*

17 Theodore Charitos (UU) *Reasoning with Dynamic Networks in Practice*

18 Bart Orriens (UvT) *On the development and management of adaptive business collaborations*

19 David Levy (UM) *Intimate relationships with artificial partners*

20 Slinger Jansen (UU) *Customer Configuration Updating in a Software Supply Network*

21 Karianne Vermaas (UU) *Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005*

22 Zlatko Zlatev (UT) *Goal-oriented design of value and process models from patterns*

23 Peter Barna (TU/e) *Specification of Application Logic in Web Information Systems*

24 Georgina Ramírez Camps (CWI) *Structural Features in XML Retrieval*

25 Joost Schalken (VU) *Empirical Investigations in Software Process Improvement*

2008

1 Katalin Boer-Sorbán (EUR) *Agent-Based Simulation of Financial Markets: A modular, continuous-time approach*

2 Alexei Sharpanskykh (VU) *On Computer-Aided Methods for Modeling and Analysis of Organizations*

3 Vera Hollink (UvA) *Optimizing hierarchical menus: a usage-based approach*

4 Ander de Keijzer (UT) *Management of Uncertain Data - towards unattended integration*

5 Bela Mutschler (UT) *Modeling and simulating causal dependencies on process-aware information systems from a cost perspective*

6 Arjen Hommersom (RUN) *On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective*

7 Peter van Rosmalen (OU) *Supporting the tutor in the design and support of adaptive e-learning*

8 Janneke Bolt (UU) *Bayesian Networks: Aspects of Approximate Inference*

9 Christof van Nimwegen (UU) *The paradox of the guided user: assistance can be counter-effective*

10 Wauter Bosma (UT) *Discourse oriented Summarization*

11 Vera Kartseva (VU) *Designing Controls for Network Organizations: a Value-Based Approach*

12 Jozsef Farkas (RUN) *A Semiotically oriented Cognitive Model of Knowledge Representation*

13 Caterina Carraciolo (UvA) *Topic Driven Access to Scientific Handbooks*

14 Arthur van Bunningen (UT) *Context-Aware Querying; Better Answers with Less Effort*

15 Martijn van Otterlo (UT) *The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains*

16 Henriette van Vugt (VU) *Embodied Agents from a User's Perspective*

17 Martin Op't Land (TUD) *Applying Architecture and Ontology to the Splitting and Allying of Enterprises*

18 Guido de Croon (UM) *Adaptive Active Vision*

19 Henning Rode (UT) *From document to entity retrieval: improving precision and performance of focused text search*

20 Rex Arendsen (UvA) *Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met een overheid op de administratieve lasten van bedrijven*

21 Krisztian Balog (UvA) *People search in the enterprise*

22 Henk Koning (UU) *Communication of IT-architecture*

23 Stefan Visscher (UU) *Bayesian network models for the management of ventilator-associated pneumonia*

24 Zharko Aleksovski (VU) *Using background knowledge in ontology matching*

25 Geert Jonker (UU) *Efficient and Equitable exchange in air traffic management plan repair using spender-signed currency*

26 Marijn Huijbregts (UT) *Segmentation, diarization and speech transcription: surprise data unraveled*

27 Hubert Vogten (OU) *Design and implementation strategies for IMS learning design*

28 Ildiko Flesh (RUN) *On the use of independence relations in Bayesian networks*

29 Dennis Reidsma (UT) *Annotations and subjective machines- Of annotators, embodied agents, users, and other humans*

30 Wouter van Atteveldt (VU) *Semantic network analysis: techniques for extracting, representing and querying media content*

31 Loes Braun (UM) *Pro-active medical information retrieval*

32 Trung B. Hui (UT) *Toward affective dialogue management using partially observable markov decision processes*

33 Frank Terpstra (UvA) *Scientific workflow design; theoretical and practical issues*

34 Jeroen de Knijf (UU) *Studies in Frequent Tree Mining*

35 Benjamin Torben-Nielsen (UvT) *Dendritic morphology: function shapes structure*

2009

1 Rasa Jurgelenaite (RUN) *Symmetric Causal Independence Models*

2 Willem Robert van Hage (VU) *Evaluating Ontology-Alignment Techniques*

3 Hans Stol (UvT) *A Framework for Evidence-based Policy Making Using IT*

4 Josephine Nabukenya (RUN) *Improving the Quality of Organisational Policy Making using Collaboration Engineering*

5 Sietse Overbeek (RUN) *Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality*

6 Muhammad Subianto (UU) *Understanding Classification*

7 Ronald Poppe (UT) *Discriminative Vision-Based Recovery and Recognition of Human Motion*

8 Volker Nannen (VU) *Evolutionary Agent-Based Policy Analysis in Dynamic Environments*

9 Benjamin Kanagwa (RUN) *Design, Discovery and Construction of Service-oriented Systems*

10 Jan Wielemaker (UvA) *Logic programming for knowledge-intensive interactive applications*

11 Alexander Boer (UvA) *Legal Theory, Sources of Law & the Semantic Web*

12 Peter Massuthe (TU/e, Humboldt-Universtät zu Berlin) *Operating Guidelines for Services*

13 Steven de Jong (UM) *Fairness in Multi-Agent Systems*

14 Maksym Korotkiy (VU) *From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)*

15 Rinke Hoekstra (UvA) *Ontology Representation - Design Patterns and Ontologies that Make Sense*

16 Fritz Reul (UvT) *New Architectures in Computer Chess*

17 Laurens van der Maaten (UvT) *Feature Extraction from Visual Data*

18 Fabian Groffen (CWI) *Armada, An Evolving Database System*

19 Valentin Robu (CWI) *Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets*

20 Bob van der Vecht (UU) *Adjustable Autonomy: Controling Influences on Decision Making*

21 Stijn Vanderlooy (UM) *Ranking and Reliable Classification*

22 Pavel Serdyukov (UT) *Search For Expertise: Going beyond direct evidence*

23 Peter Hofgesang (VU) *Modelling Web Usage in a Changing Environment*

24 Annerieke Heuvelink (VU) *Cognitive Models for Training Simulations*

25 Alex van Ballegooij (CWI) *"RAM: Array Database Management through Relational Mapping*"

26 Fernando Koch (UU) *An Agent-Based Model for the Development of Intelligent Mobile Services*

27 Christian Glahn (OU) *Contextual Support of social Engagement and Reflection on the Web*

28 Sander Evers (UT) *Sensor Data Management with Probabilistic Models*

29 Stanislav Pokraev (UT) *Model-Driven Semantic Integration of Service-Oriented Applications*

30 Marcin Zukowski (CWI) *Balancing vectorized query execution with bandwidth-optimized storage*

31 Sofiya Katrenko (UvA) *A Closer Look at Learning Relations from Text*

32 Rik Farenhorst and Remco de Boer (VU) *Architectural Knowledge Management: Supporting Architects and Auditors*

33 Khiet Truong (UT) *How Does Real Affect Affect Affect Recognition In Speech?*

34 Inge van de Weerd (UU) *Advancing in Software Product Management: An Incremental Method Engineering Approach*

35 Wouter Koelewijn (UL) *Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling*

36 Marco Kalz (OUN) *Placement Support for Learners in Learning Networks*

37 Hendrik Drachsler (OUN) *Navigation Support for Learners in Informal Learning Networks*

38 Riina Vuorikari (OU) *Tags and self-organisation: a metadata ecology for learning resources in a multilingual context*

39 Christian Stahl (TUE, Humboldt-Universität zu Berlin) *Service Substitution – A Behavioral Approach Based on Petri Nets*

40 Stephan Raaijmakers (UvT) *Multinomial Language Learning: Investigations into the Geometry of Language*

41 Igor Berezhnyy (UvT) *Digital Analysis of Paintings*

42 Toine Bogers (UvT) *Recommender Systems for Social Bookmarking*

TiCC Dissertation Series

1. Pashiera Barkhuysen
Audiovisual Prosody in Interaction
Promotores: M.G.J. Swerts, E.J. Krahmer
Tilburg, 3 October 2008

2. Ben Torben-Nielsen
Dendritic Morphology: Function Shapes Structure
Promotores: H.J. van den Herik, E.O. Postma
Co-promotor: K.P. Tuyls
Tilburg, 3 December 2008

3. Hans Stol
A Framework for Evidence-based Policy Making using IT
Promotor: H.J. van den Herik
Tilburg, 21 January 2009

4. Jeroen Geertzen
Act Recognition and Prediction. Explorations in Computational Dialogue Modelling
Promotor: H.C. Bunt
Co-promotor: J.M.B. Terken
Tilburg, 11 February 2009

5. Sander Canisius
Structural Prediction for Natural Language Processing: A Constraint Satisfaction Approach
Promotores: A.P.J. van den Bosch, W.M.P. Daelemans
Tilburg, 13 February 2009

6. Fritz Reul
New Architectures in Computer Chess
Promotor: H.J. van den Herik
Co-promotor: J.H.W.M. Uiterwijk
Tilburg, 17 June 2009

7. Laurens van der Maaten
Feature Extraction from Visual Data
Promotores: E.O. Postma, H.J. van den Herik
Co-promotor: A.G. Lange
Tilburg, 23 June 2009

8. Stephan Raaijmakers
Multinomial Language Learning: Investigations into the Geometry of Language
Promotor: A.P.J. van den Bosch
Tilburg, 1 December 2009

9. Igor Berezhnyy
Digital Analysis of Paintings
Promotores: E.O. Postma, H.J. van den Herik
Tilburg, 7 December 2009

10. Toine Bogers
Recommender Systems for Social Bookmarking
Promotor: A.P.J. van den Bosch
Tilburg, 8 December 2009