Fusing Recommendations for Social Bookmarking Websites

Toine Bogers[a], Antal van den Bosch[b]

[a]*Royal School of Library and Information Science, Birketinget 6, DK-2300, Copenhagen, Denmark*
[b]*Tilburg centre for Communication and Cognition, Tilburg University, NL-5000, Tilburg, the Netherlands*

Email addresses: tb@db.dk (Toine Bogers), Antal.vdnBosch@uvt.nl (Antal van den Bosch)

Fusing Recommendations for Social Bookmarking Websites

Abstract

Social bookmarking websites, which allow their users to store and access their bookmarks online through a Web interface, are rapidly growing in popularity. Recommender systems are a successful remedy to the information overload accompanying the popularity-driven explosive growth in content. They are designed to automatically identify which unseen content might be of interest to a particular user, based on his or her past preferences. In this article, we focus on the task of item recommendation: using recommender systems to locate interesting content for users of social bookmarking websites. Item recommendation for social bookmarking has only recently begun to attract more attention from researchers, and much of the previous work suffers from a lack of comparisons between the different available approaches. This means that it is difficult to determine exactly what the best practices are for item recommendation on social bookmarking websites. In this article, we address this issue by comparing and evaluating eight different recommendation approaches on four different data sets from two different domains. We find that approaches that use tag overlap and metadata are competitive with each other and provide better results for social bookmarking data sets than the transaction patterns that are used traditionally in recommender systems research. In addition, we investigate how to fuse together different recommendation approaches to further improve recommendation accuracy. We find that fusing recommendations can indeed produce significant improvements in recommendation accuracy. We also find that it is often better to combine approaches that use different data representations, such as tags and metadata, than it is to combine approaches that only vary in the algorithms they use. The best results are obtained when both of these aspects of the recommendation task are varied in the fusion process. Our findings can be used to improve the quality of recommendations not only on social bookmarking websites, but conceivably also on websites that offer annotated commercial content.

Key words: Social bookmarking, recommender systems, data fusion, collaborative filtering, content-based filtering

1. Introduction

Arguably, *social bookmarking websites* owe their rapidly growing popularity as a social media application to their emphasis on open collaborative information access. Offering an alternative to keeping local copies of pointers to favorite URLs, social bookmarking websites allow their users to 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[1].

In addition to the aforementioned storage and management functionality, most social bookmarking websites also offer the user the opportunity to describe the content they have added to their personal library by keywords. These keywords are commonly referred to as *tags* and can be selected freely by the user. 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 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, bookmarked items, and tags, commonly referred to as a *folksonomy* (Vander Wal, 2005). This phenomenon of assigning tags is also known as *social tagging*, and the resulting folksonomies have become a staple of many Web 2.0 websites and services (Golder and Huberman, 2006).

[1] In the remainder of this article we will use the term 'social bookmarking' for both types of social storage and management services.

Information Access on Social Bookmarking Websites The success of social bookmarking websites depends partly on how these connections between users, items, and tags are exploited to improve the user's access to his[2] own bookmarks and those of others. Typically, users can browse the social bookmarking websites folksonomy in all directions to view not only their own bookmarks, but also browse through them using a so-called *tag cloud*, an alphabetically sorted list of all tags with some sort of visual markup denoting a tag's popularity, based on, e.g., its frequency of use. In addition, users can also view an item's history, i.e., all occurrences of that item being bookmarked by a user; and the history of a tag, i.e., all the other bookmarks that have used that item to annotate a bookmark. Figure 1 illustrates these possibilities for Delicious[3], one of the most popular social bookmarking websites.

However, the rapid growth in popularity of social bookmarking websites is accompanied by a commensurate growth of content on these websites, making the browsing mechanisms illustrated in Figure 1 less effective over time. Different information access technologies exist that can cope with this problem. These technologies can be grouped in *user-initiated* and *system-initiated*, differing in the type of effort required by the user.

User-initiated information access technologies require the user to take action to find relevant or interesting content. A typical example is browsing. To alleviate the problem of increasing ambiguity caused by information overload, some people have suggested automatic clustering and disambiguation methods to better guide the user in the browsing process (Li et al., 2007; Clements et al., 2008b). Search engines are a second type of user-initiated information access technology, where the user actively has to formulate his information need before the system can attempt to locate and rank the most relevant content. Most social bookmarking websites offer only rudimentary search functionality, so several approaches have been proposed in recent years that specifically target social bookmarking search (Bao et al., 2007; Zhou et al., 2008; Carman et al., 2008).

System-initiated information access technologies attempt to automatically locate interesting content on the user's behalf. Recommender systems are a typical example of such technology. They belong to a class of per-

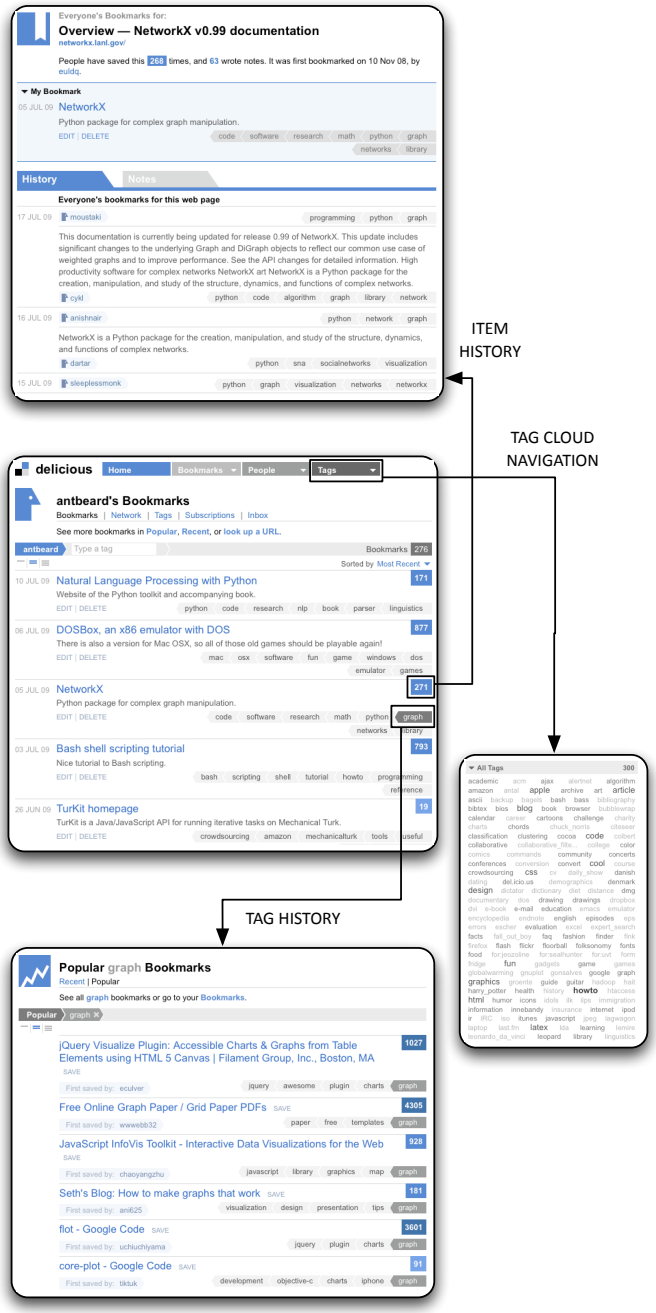

Figure 1: Navigation on a social bookmarking website. The starting point for every user is their profile page which lists their bookmarks (top left). From there, users can browse to tag pages (bottom left) which show which other bookmarks have been tagged with the selected tag; and to item history pages (bottom right), which show all other users who have posted the selected item, and what tags and metadata they assigned to it. Users can also get an overview of their tags through the tag cloud view (top right), which marks up the more frequently used tags with a darker font color and a larger font size.

[2]In this article we use 'his' and 'he' to refer to both genders.
[3]http://www.delicious.com/

sonalized information filtering technologies that aim to identify which items in a catalog might be of interest to a particular user. Recommendations can be made using a variety of information sources related to both the user and the items: past user preferences, purchase history, demographic information, item popularity, the metadata characteristics of the products, etc. While we believe that all three approaches, browsing, search, and recommendation, are essential for unlocking the potential of a social bookmarking system, we focus solely on recommender systems in this article.

The most popular application of recommendation technology to social bookmarking so far has been tag recommendation: suggesting appropriate tags to a user when posting a new bookmark to the system. Many approaches have been proposed in the past years (Xu et al., 2006; Jäschke et al., 2007; Heymann et al., 2008; Song et al., 2008). Yet, tag recommendation helps users only when they post new content to the system, not when they are interested in content other users have posted. To address this issue, recommender systems can also be used to generate content recommendations to be presented at the user at any moment. This task is the focus of this article.

Item Recommendation for Social Bookmarking The task of *item recommendation* involves retrieving and recommending interesting and relevant items—in our study, bookmarks or scientific articles—to the user. Recommendations can be based on a variety of information sources about the user and the items, offering information at different representation levels. One level of representation, *usage data*, is the set of transaction patterns that shows which items have been posted by a user, and which users have posted an item. In social bookmarking, tags offer an additional level of representation, linking users to items through an alternative route. The past couple of years have seen a growing number of approaches for item recommendation that incorporate these two types of data representations (Hotho et al., 2006a; Clements et al., 2008a; Tso-Sutter et al., 2008; Wetzker et al., 2009). These approaches typically use a *collaborative filtering* (CF) algorithm to produce their recommendations. CF attempts to emulate "word-of-mouth" recommendations. Here, items are recommended to users based a profile of their previous activities, what items like-minded users have posted to their profiles, and how they were tagged.

Social bookmarking websites also enable users to attach item-specific metadata to their items, such as the item title, summary, author information, etc. This additional metadata is a third type of data representation that could be used to support the recommendation process. Using item metadata or even the entire digital content of an item to improve recommendation quality is called *content-based filtering* (CBF). Using metadata or content representations to support recommendation is not new (Alspector et al., 1997). However, its application to social bookmarking websites is rare. A first investigation into the use of such item metadata for recommending content on social bookmarking websites was reported by Bogers and Van den Bosch (2009).

In the early stages of the field of recommendation technologies it is difficult to obtain an overview of best practices for item recommendation. 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. There is also a lack of a common evaluation methodology, as many researchers construct and motivate their own evaluation metric. Finally, there have been virtually 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. This article consists of two main parts. In the first part of this article, in Section 3, we address a number of these criticisms by performing a systematic comparison of a number of different CF and CBF recommendation algorithms using a well-established evaluation metric on four data sets of different sizes and domains. We show that while certain algorithms tend to outperform others, there is no clear advantage to using CF approaches over CBF approaches or vice versa.

Fusing Recommendations While a fair and balanced comparison of different types of recommendation algorithms for social bookmarking is a necessary step, we believe that the problem of effective item recommendation is too complex for any individual solution to capture in its entirety. We expect that by combining different approaches we can produce better recommendations. In the second part of this article, in Section 4, we examine the possibilities of *data fusion* of different recommendation approaches for social bookmarking websites in more detail. We will examine the effectiveness of combining the output of eight different recommen-

dation runs[4].

Contributions The contributions of this article are twofold. First, we compare a number of different algorithms from both the CF and CBF recommendation approaches, and determine which work best for the task of item recommendation on social bookmarking websites. We perform our experiments using a well-established evaluation metric on four data sets of different sizes. These data sets cover two different domains, Web bookmarks and scientific articles, and all of them are publicly available.

Second, we determine the most effective way of combining different recommendation algorithms. We believe that the problem of item recommendation cannot be solved by a single one-size-fits-all approach. We show that by combining different approaches that model different aspects of this problem, we can produce better recommendations. We consider two aspects of the recommendation task: the data representation and the recommendation algorithm. Using these two aspects, we perform a systematic comparison of different combination techniques, and show that we can achieve significantly better results by combining approaches that cover different aspects.

Our overall findings with regard to recommendation are directly applicable to social bookmarking websites. They could also be used to improve recommender systems on other websites that offer a large catalog of items annotated with metadata and tagged by their users, of the type of Amazon.com.

This article is structured as follows. In Section 2 we describe the setup of our experiments, such as data sets, filtering regimen, and evaluation metrics used. We describe and compare eight item recommendation algorithms in Section 3. In Section 4 we then describe how we combine these individual algorithms, and present the results of our fusion experiments as well as an analysis of our findings. Related work relevant to item recommendation and recommender systems fusion is discussed in their respective sections. We formulate our conclusions in Section 5.

2. Experimental Setup

In this section, we describe the experimental setup for our recommendation and fusion experiments. We perform our experiments using a well-established evaluation metric on four data sets of different sizes. These data sets cover two different domains, Web bookmarks and scientific articles, and three of them are publicly available. We describe these data sets in more detail in Subsection 2.1. In Subsection 2.2, we describe how we filtered out noise from our data set, and in Subsection 2.3 we discuss our evaluation setup.

2.1. Data Sets

We base our experiments on four data sets that were collected from three different social bookmarking websites with different characteristics: CiteULike, BibSonomy, and Delicious. Two data sets correspond to the domain of Web page bookmarks (Delicious and BibSonomy) and the other two cover the domain of scientific articles (Delicious and BibSonomy). 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. All four data sets have been made publicly available to the recommender systems community[5].

CiteULike is a social bookmarking service that allows its users to add their academic reference library to an online profile[6]. Articles can be stored with their metadata, abstracts, and links to the papers at the publishers' websites. Users can also add personal comments and tags. CiteULike offers free access to daily dumps of their core database. 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. It does not, however, contain any other item metadata, so we crawled this ourselves from the CiteULike website using the article IDs. Articles are annotated using the standard BibTeX-like fields, such as title, author names, page numbers, publisher information, etc. We were able to crawl metadata for 72.8% of all articles in the November 2007 dump, resulting in 18,072 user profiles.

BibSonomy is a social bookmarking service for sharing Web page bookmarks and reference lists of scientific articles[7]. Items are stored and represented by their BibTeX metadata representations. These can include

[4] We use the term *run* to refer to the output of a recommendation algorithm, a ranked list of recommended items for each active user.

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

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

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

Table 1: Statistics of the filtered versions of our four data sets.

	bookmarks		articles	
	Delicious	**BibSonomy**	**CiteULike**	**BibSonomy**
# users	1,243	192	1,322	167
# items	152,698	11,165	38,419	12,982
# tags	42,820	13,233	28,312	5,165
# posts	238,070	29,096	84,637	29,720
user-item sparsity (%)	99.8746	98.6427	99.8334	98.6291
avg # items per user	191.5	151.5	64.0	178.0
avg # users per item	1.6	2.6	2.2	2.3
avg # tags per user	192.1	203.3	57.3	79.2
avg # users per tag	5.6	2.9	2.7	2.6
avg # tags per item	4.8	8.4	5.3	3.1
avg # items per tag	17.0	7.1	7.3	7.7

abstracts and links to the papers at the publishers' websites. Users are able to tag their bookmarked content and use these tags to browse and discover related references (Hotho et al., 2006b). BibSonomy's creators organized the 2008 ECML/PKDD Discovery Challenge which focused on social bookmarking, and released the BibSonomy data set to the public in May 2008 as part of this challenge[8]. The organizers made available a snapshot of the BibSonomy system, consisting of all resources posted to BibSonomy between its inception in 2006 and March 31, 2008. It includes the same type of article metadata as we collected for 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 (Bibsonomy Bookmarks), and a data set containing only scientific articles (Bibsonomy Articles). The snapshot contains 1,913 user profiles containing web page bookmarks and 1,310 user profiles containing scientific articles.

Delicious is a social bookmarking service for storing, sharing, and discovering web bookmarks. It allows its users to manage and tag URLs of web pages[9]. Unlike CiteULike and BibSonomy, Delicious does not offer data dumps of their databases, so we gathered our data set by crawling a subset of the Delicious website. Because of our focus on the task of item recommendation for users, our aim was to collect a balanced, unbiased set of user profiles, i.e. the complete set of bookmarks a user had posted to Delicious. From an earlier breadth-first crawl of Delicious we obtained a list of 300,000 users. We randomly selected around 18,000 of these users to match the size of our CiteULike data set, and crawled the complete profiles of these users.

2.2. Data Set Filtering

It is common practice in recommender system evaluation to select realistic subsets of the data sets used to ensure that reliable recommendations can be generated. This also allows for a fair comparisons of different recommendation algorithms (Herlocker et al., 2004). This is typically done by filtering out users or items whose profile size or popularity falls below a certain threshold. We follow this procedure in our preparation of the data sets as well. We only retain the users who have added 20 items or more to their personal profile. Table 1 lists the statistics of our four data sets after filtering.

2.3. Evaluation Setup & Metrics

We evaluate our algorithms on the "Find Good Items" recommendation task, also known as Top-*N* recommendation, where users are provided with a ranked list of recommended items based on their personal profile (Herlocker et al., 2004). We divide each data set into a training and test set by randomly selecting 10% of the users to be in our test set. Final performance is evaluated on this 10% by withholding 10 items from each of these so-called *active users*, and using the remaining profile items together with the training set to generate the recommendations for those 10%. If the withheld items are predicted at the top of the ranked result list, then the algorithm is considered to perform well. To prevent overestimation when optimizing algorithm parameters, we use 10-fold cross-validation. We subdivide our training set into 10 folds and use these

[8] http://www.kde.cs.uni-kassel.de/ws/rsdc08/
[9] http://www.delicious.com/

for 10-fold cross-validation of our parameter optimization. For each fold, 10 items are withheld from the test fold users, to be retrieved by the recommendation algorithm. The final values for our evaluation metric on the withheld items are then averaged over the 10 folds.

In our evaluation, we adopt an IR perspective by treating each of the users as a separate query or topic. The 10 withheld items for each user constitute the items for which we have relevance judgments. For each user, a ranked list of items is produced and evaluated on whether these withheld items show up in the result list. This approach is also known as *backtesting*. While it is certainly possible that the recommendation lists contain recommendations that the user would find relevant or interesting, we cannot know this without the user judging them. Herlocker et al. (2004) assess the usefulness of different metrics for different types of recommendation tasks. For the Top-N recommendation task, they find that metrics that take into account the ranking of the items are most appropriate. We therefore evaluate our algorithms using Mean Average Precision (MAP), which is defined as the average of the Average Precision values calculated per relevant retrieved item. For determining the significance of differences between runs we use a two-tailed paired Student's t-test. We report on significant differences against the best baseline runs using $^{\triangle}$ (and $^{\triangledown}$) for $\alpha = .05$ and $^{\blacktriangle}$ (and $^{\blacktriangledown}$) for $\alpha = .01$. For instance, a $^{\triangle}$ signals a significant improvement of, for instance, a fusion run over the best-performing individual component run at $\alpha = .05$.

3. Item Recommendation for Social Bookmarking

In Section 1, we mentioned three criticisms of previous work on item recommendation for social bookmarking: (1) a lack of common data sets, (2) the use of different evaluation metrics, and (3) no within-study comparisons of different recommendation algorithms. Using our experimental setup, we aim to address these criticisms in this section by comparing eight different methods of generating item recommendations for social bookmarking systems.

Social bookmarking systems offer two specific types of information sources that can be put to use when generating item recommendation: (1) the folksonomy and (2) the metadata assigned to items and their (textual) content. We examine the usefulness of both information sources for recommendation in this section. We discuss four different CF algorithms in Subsection 3.1 that use the folksonomy, and four different metadata-based algorithms in Subsection 3.2. Note that for the second type we restrict ourselves to using metadata only, as it was impractical or impossible to collect all item content for our data sets. We describe and analyze the results of our comparisons in Subsection 3.3. We conclude this section with an overview of the related work in Subsection 3.4.

3.1. Collaborative Filtering

In Section 1, we defined the folksonomy of a social bookmarking website as an extra annotation layer connecting users to items. With this extra layer we can extract two types of representations of patterns connecting users to items. The first is *usage data*, which is the set of transaction patterns that shows which items have been posted by a user, and which users have posted an item. These usage patterns are a common and well-understood source of information for recommendation. The second is an indirect route of *user–tag* and *tag–item assignment patterns*. User–tag patterns characterize a user's interest similar to user–item patterns, while tag–item patterns characterize the interest of an aggregate of users in the item. Both routes allow us to estimate similarities between pairs of users or items.

The class of algorithms that exploit these types of representations for recommendation purposes are referred to as collaborative filtering (CF) algorithms. We focus on using and extending one specific CF algorithm: the k-Nearest Neighbor (k-NN) algorithm. We pick k-NN as it is a well understood algorithm that can easily be extended to include other information in addition to transaction patterns (Herlocker et al., 1999; Burke, 2002). There are two flavors of the k-NN algorithm for CF: *user-based CF* and *item-based CF*.

User-based Collaborative Filtering The k-NN algorithm for user-based CF consists of two steps. First, we locate the users that are most similar to the *active user*, i.e., the user we are trying to recommend new items to. This means we calculate the similarity between the active user and all other users in the system. One way of determining the similarity between a pair of users is looking at usage data and considering the overlap in items they have posted to their profile. We represent each user u_k as a unary user profile vector $\mathbf{u}_k$ that represents all the items that were posted by u_k with a 1. We use the cosine similarity metric to determine the similarity between two users u_k and the active user u_a as $sim_{cosine}(u_a, u_k) = \frac{\mathbf{u}_a \cdot \mathbf{u}_k}{||\mathbf{u}_a|| \, ||\mathbf{u}_k||}$. Cosine similarity has been

used successfully with data sets with implicit ratings (Breese et al., 1998).

In the second step we gather the items of the most like-minded users to determine which would be suitable recommendations for the active user. The assumption here being that the more similar two users are in the items they share, the more like-minded they are. We only consider the top k most similar users for the active user u_a as the Set of Similar Users $SSU(u_a)$. Using this set of nearest neighbors we generate the final prediction scores $\widehat{x}_{a,l}$ for each of the SSU's items i_l as $\widehat{x}_{a,l} = \sum_{u_k \in SSU(u_a)} sim_{cosine}(u_a, u_k)$. Thus, the predicted score of an item i_l is the sum of the similarity values (between 0 and 1) of all N nearest neighbors that actually posted item i_l. Finally, all items are ranked by their predicted score $\widehat{x}_{k,l}$. Items already posted by the active user are filtered out to produce the final list of recommendations for the active user.

Instead of using the usage data to calculate user similarity, we can determine the similarity between a pair of users by considering the overlap in tags they have assigned to their items. In this case, the user profile vector $\mathbf{u}_k$ lists all the tags that were used by u_k with their frequency. When we calculate user similarity using these tag vectors, we can generate recommendations in the same way as with usage data: first, calculate the similarity between the active user and all other users, followed by using the top k most like-minded users to generate item recommendations.

Item-based Collaborative Filtering The item-based k-NN algorithm operates analogously to the user-based filtering algorithm, but focuses on item–item similarity instead of user–user similarity (Sarwar et al., 2001). Instead of comparing users directly, we try to identify the best recommendations for each of the items in an active user's profile. In other words, for item-based filtering we calculate the similarities between the training items of the active user u_a—his *active items*—and the other items that user has not yet posted. This similarity is based on the overlap in users that have posted the two items. We represent each item i_l as a unary item profile vector $\mathbf{i}_l$ that lists all the users that posted i_l with a 1. We use the cosine similarity to determine the similarity between two items i_l and i_b as $sim_{cosine}(i_l, i_b)$.

Next, we identify the top k most similar items for each of the active user's items i_b separately. We define this neighborhood as the Set of Similar Items $SSI(i_b)$, where we select the top k of all items not already in the active user's profile, ranked by their cosine similarity to item i_b. Using this set of nearest neighbors we generate the final prediction score $\widehat{x}_{a,b}$ for each unseen item i_b as $\sum_{i_l \in SSI(i_b)} sim_{cosine}(i_b, i_l)$. This is repeated for each of the user's active items i_b. Here, the predicted score is the sum of the similarity values (between 0 and 1) of all the most similar items that were posted by user u_a.

As in user-based CF, instead of using the usage data to calculate item similarity, we can also determine the similarity between a pair of items by considering the overlap in tags they were annotated with by all users. In this case, the item profile vector $\mathbf{i}_l$ lists all the tags that were assigned to i_l with their frequency. The recommendation algorithm is identical to when usage data is employed: first, the similarities between the active items' and all other items are determined, followed by using the top k most similar items for each active item to generate the final recommendations.

For all four CF approaches, the top k neighbors are used to generate the recommendations. The k hyperparameter can influence prediction quality significantly. 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 therefore use our 10-fold cross-validation setup to optimize the number of neighbors N as described in Subsection 2.3.

3.2. Metadata-based Recommendation

In addition to the folksonomic structure of the underlying network, social bookmarking services also offer users the possibility to annotate the content of their items with metadata. The standard way of incorporating metadata in CBF is to use it to represent the content in a system. Item content representations can then be matched against the active user's profile to find the items that are most relevant to that user. We propose two content-based filtering algorithms that directly match the metadata assigned by active users with the metadata of all other items in a single step.

We also explore another perspective on metadata by seeing it as yet another source for calculating user and item similarities in addition to the usage data and tags from Subsection 3.1. We can then plug the newly calculated user and item similarities into the standard k-NN algorithm. The resulting algorithm is a hybrid of CF and CBF techniques, and we refer to this perspective as *hybrid filtering*. Before we move on to describing these four algorithms in Subsections 3.2.1 through 3.2.3, we

first take a closer look at the metadata we have available in our data sets and how we selected the metadata fields to use in our experiments.

Selecting Metadata Although all social bookmarking services allow their users 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 allow more metadata to be added, which reflects the more complex creation and publication process of scientific papers.

We distinguish between *item-intrinsic* and *item-extrinsic* metadata. Item-intrinsic metadata fields relate directly to the content of the item being annotated. For the two data sets dealing with web bookmarks these include DESCRIPTION, TAGS, TITLE, and URL. The two scientific article data sets contain the additional intrinsic fields ABSTRACT, AUTHOR, BOOKTITLE, EDITOR, JOURNAL, NOTE, and SERIES. Table 2 lists the item-intrinsic metadata available for our four data sets.

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 CHAPTER, MONTH, or PAGES—cannot be expected to directly generate appropriate recommendations. 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 will have metadata fields that fall into one of these two categories, and that are easily identifiable as such.

These sets of metadata fields allow for many experiments with subsets of metadata fields. However, we observed in our experiments that, in general, the best recommendations are produced by using all of the item-intrinsic metadata fields combined. Although certain individual item-intrinsic metadata fields can give good performance, such as AUTHOR, DESCRIPTION, TAGS, and TITLE, they rarely outperform the combination of all item-intrinsic metadata fields. Including item-extrinsic metadata fields never yielded a significant improvement in performance. We therefore only report on the results of experiments that uses item-intrinsic metadata fields in this paper.

Table 2: An overview of the metadata available in our four data sets.

Domain type	Metadata fields
Web bookmarks (Delicious, BibSonomy)	DESCRIPTION, TAGS, TITLE URL
Scientific articles (CiteULike, BibSonomy)	ABSTRACT, AUTHOR, BOOKTITLE, DESCRIPTION, EDITOR, JOURNAL, NOTE, SERIES, TAGS, TITLE, URL

3.2.1. *Content-based Filtering*

In content-based filtering, the focus 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 to rank-order the unseen items by their similarity to the interest profile, thereby approximating potential interest in those items. Figure 2 illustrates two different algorithms we propose for content-based filtering: *profile-centric matching* and *post-centric matching*.

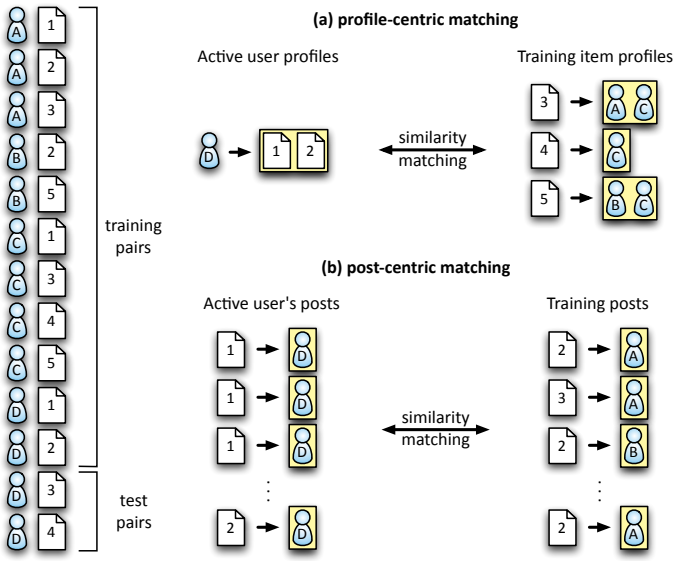

Figure 2: Visualization of our two content-based filtering approaches to item recommendation for a small toy data set.

Profile-centric Matching The difference between our two CBF algorithms is the level of aggregation. In our profile-centric matching approach, as illustrated in the top half of Figure 2, we aggregate all of the metadata of

the active user's items into a single user profile. The intuition here is that by doing this we can completely capture the user's interests. By analogy, we can construct the profile of an item i_l by gathering and combining all of the metadata assigned to i_l by users in the training set. We then match each active user profile against all item profiles for items new to the active users to produce a ranking of all items. After removing the items already in the active user's profile, we are left with the final rank-ordered list of recommendations.

The left half of Figure 2 visualizes a 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 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 posted so far (items 1 and 2). Then the item profiles are generated 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 carry the most similar metadata. Our similarity matching technique is the same for all four metadata-based approaches; we explain it in more detail in Subsection 3.2.3.

Post-centric Matching In contrast to profile-centric matching, *post-centric matching* operates on the level of individual *posts*. We match each of an active user's posts separately against all the other posts of unseen items in the training set. This leads to a list of matching posts in order of similarity for each of the active user's posts. Since retrieval scores are not directly comparable between runs, we normalize the original similarity scores sim_{org} into $[0, 1]$ using the maximum and minimum similarity scores sim_{max} and sim_{min} according to $sim_{norm} = \frac{sim_{org} - sim_{min}}{sim_{max} - sim_{min}}$. We then calculate a rank-corrected sum of similarity scores for each item i_l according to $score(i) = \sum \frac{sim_{norm}(i_l)}{log(rank(i_l))+1}$. The final list of recommendations ranks every unseen item i_l by their rank-corrected score $score(i_l)$.

Post-centric matching is illustrated in the bottom half of Figure 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.

3.2.2. Hybrid Filtering

In addition to focusing solely on using the metadata for recommendation, we also consider a hybrid approach that joins CBF and CF. Combining the two 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 of users and items in CF algorithms. Figure 3 illustrates this approach. Instead of only looking at the overlap in items that two users have in common when calculating user similarities, we can use the textual overlap in the metadata applied to items to determine the most similar neighbors. Users that describe their profile items using the same terminology are likely share the same interests, making them a good source of recommendations. This is similar to the way we used the tag clouds of users and items to calculate similarity between users and items in Subsection 3.1. Similarly, items that share much of the same metadata are more likely to be similar. The user–user and item–item similarities we derive in this way are then plugged into the standard k-NN CF algorithm to produce the item recommendations. The resulting algorithm is similar to a CF algorithm, but with a metadata-based data representation.

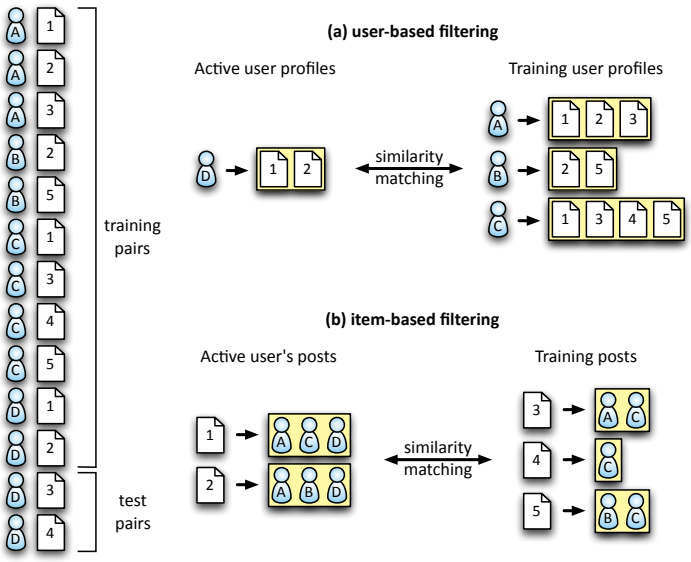

Figure 3: Visualization of our two hybrid filtering approaches to item recommendation for a small toy data set.

Hybrid filtering also consists of two steps: (1) calculating the most similar neighbors of the active user or his items, and (2) using those neighbors to predict item ratings for the active user. The latter prediction step is performed in the same manner as described earlier in Section 3.1. As in CF, with our hybrid filtering algorithms we also distinguish between user-based filtering, where we generate recommendations by determining the most similar users, and item-based filtering, where we recommend the items most similar to the items in the active user's profile. Like in Section 3.2.1, we approach the first step from an IR perspective and calculate the textual similarities between users or items. For each user and each item we generate user and item profile representations, constructed as follows. All of the metadata text of a user's posts is collated into a single user profile for that user. Similarly, for the item-based approach 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.

3.2.3. Similarity Matching

We calculate the similarity between two profiles, regardless of their level of granularity, by measuring the textual overlap between the metadata contained in those profiles. We approach this from an IR perspective and restrict ourselves to measuring textual similarity. We use version 2.7 of the open-source retrieval toolkit Lemur to calculate the similarities between the different user and item profiles. The Lemur toolkit[10] implements different retrieval methods based on language modeling (Strohman et al., 2005). Preliminary experiments comparing language modeling with the OKAPI model and a tf·idf approach suggested a language modeling approach with Jelinek-Mercer smoothing as the best-performing similarity matching method. The language models we used are maximum likelihood estimates of the unigram occurrence probabilities. We filter stopwords using the SMART stopword list and do not perform stemming.

3.3. Results & Analysis

Collaborative Filtering Table 3 shows the results of our four CF runs and our four metadata-based runs. The top half of the table shows the results of the four CF runs, two runs based on usage data and two based on tagging data. We observe a slight edge of the user-based k-NN algorithm over the item-based variant, 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, so there is no clear winner here.

As for the results with tag overlap, we observe that item similarities based on tag overlap work well for item-based filtering. If we compare the scores of the item-based CF algorithm with tagging similarity to those with usage data similarity, we can we see considerable improvements over the best usage-based CF runs. Performance increases range from 49% on Bibsonomy Articles to almost 274% on Delicious, but these are only statistically significant on the Delicious data set. We see the opposite trend for user-based filtering, where tag overlap results in significantly worse scores compared to user-based CF with usage data similarity on all data sets, with performance decreases ranging from 40% to 63%. This means that using tag overlap in item-based filtering makes item-based filtering outperform user-based filtering on all four data sets.

We believe that it is the reduction in sparsity from using tag overlap that causes this difference in performance. On average, the number of tags assigned to an item is 2.5 times higher than the number of users who have posted the item. This means that, on average, item profile vectors containing tagging information are less sparse than item profile vectors containing usage information, making the possibility of overlap between vectors more likely. Using more values in the similarity calculation leads to a better estimate of the real similarity between two items, which in turn leads to the better performance of using tag overlap for item-based CF. For user-based filtering 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.

It appears that bookmark recommendation is more difficult than article recommendation. Even with the same collection, recommending BibSonomy bookmarks achieves MAP scores that are nearly three times as low as recommending articles. Arguably, bookmark recommendation is a more difficult problem because of the open domain. We believe this is due to a differ-

[10] Available at http://www.lemurproject.org

Table 3: Results of the eight different recommendation runs. Reported are the MAP scores. Best-performing runs for each group of approaches, CF and metadata-based, are printed in bold. Boxed runs are the best overall.

Runs	bookmarks		articles	
	BibSonomy	Delicious	BibSonomy	CiteULike
User-based CF with usage data similarity	0.0277	0.0046	0.0865	0.0757
Item-based CF with usage data similarity	0.0244	0.0027	0.0737	**0.0887**
User-based CF with tag similarity	0.0102	0.0017	0.0459	0.0449
Item-based CF with tag similarity	**0.0370**	**0.0101**	**0.1100**	0.0814
Profile-centric matching	**0.0402**	0.0011	0.1279	**0.0978**
Post-centric matching	0.0259	0.0023	0.1190	0.0455
User-based CF with metadata similarity	0.0197	**0.0039**	0.0155	0.0536
Item-based CF with metadata similarity	0.0267	0.0017	**0.1510**	0.0719

ence in topic specificity. The Delicious and Bibsonomy Bookmarks data sets cover bookmarks of web pages, which encompass many more topics than scientific articles do. Users of Delicious and Bibsonomy Bookmarks can be expected to have heterogeneous topics in their profile, making it more difficult to recommend new, interesting bookmarks based on their profiles. We see evidence for this explanation in the average number of unique tags per user: 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. These average number of tags per user is 203.3 and 192.1 for Bibsonomy Bookmarks and Delicious respectively, which is markedly higher than the 79.2 and 57.3 for Bibsonomy Articles and CiteULike. This suggests that the bookmark data sets are indeed more heterogeneous in terms of topics.

Metadata-based Recommendation The bottom half of Table 3 contains the MAP scores for the four metadata-based algorithms. Looking at our two CBF algorithms, we can observe that the profile-centric approach tends to outperform the post-centric approach on three of our four data sets. This improvement is statistically significant for the CiteULike data set with an improvement of 115% ($p < 10^{-5}$). Only on the Delicious data set does post-centric matching perform significantly better ($p < 0.05$). A likely explanation for these findings is that posts carry little metadata. 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 less sparse 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.

If we look at the two hybrid filtering algorithms, we see that the item-based CF approach with metadata similarity outperforms the user-based variant on three of our four data sets. On the Bibsonomy Articles and CiteULike data sets these differences are statistically significant and especially large at 874% ($p < 0.01$) and 34% ($p < 0.05$) respectively. One might 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. This is apparently not the case, however. Item profiles are a higher-level aggregation than posts, and coverage is indeed a bit higher for 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., $k = 1$). 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 neighboring user matched on so strongly. However, in selecting the items to be recommended using user-based CF 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 k 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.

If we compare all four metadata-based algorithms, it is difficult to assign a clear winner among the algorithms. However, the profile-centric matching algorithm performs best on two data sets and is a consistently competitive algorithm on the other two data sets.

General Findings On three out of four data sets a recommendation algorithm that uses metadata is better than the best CF run using data from the folksonomy. We believe this is because using metadata results in richer representations for matching users and items. 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. While the best approach seems to be dependent on the data set and the domain, aggregating all of the intrinsic metadata at the user and item level results in algorithms that outperform the algorithms using only information from the folksonomy. With such a variety in algorithm performance, it will be interesting to see whether these best algorithms represent a ceiling in recommendation quality, or whether we can gain something by combining different algorithms. We examine this question in more detail in Section 4.

3.4. Related Work

Collaborative Filtering One of the first approaches to recommendation for social bookmarking websites was presented by Hotho et al. (2006a), who proposed a graph-based algorithm called *FolkRank*. They generated 2D projections of the tripartite graph and proposed a random walk model similar to PageRank (Page et al., 1998) that uses the steady state node probabilities as the basis for ranking their recommendations. Clements et al. (2008a) also proposed a random walk model for item recommendation, but combine ratings information with tagging information into a single model. They also incorporated self-transition probabilities in the matrix, and used the walk length as an algorithm parameter.

There have also been several adaptations of memory-based algorithms that include information about the tags assigned by users to items (Amer-Yahia et al., 2008; Nakamoto et al., 2007; Szomszor et al., 2007). Tso-Sutter et al. (2008) proposed a novel tag-aware k-NN algorithm for item recommendation. When 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. This fused model is able to effectively capture the relationship between users, items, and tags.

Symeonidis et al. (2008) were among the first to propose a model-based approach to incorporating tagging information in recommendation. They propose an item recommendation approach that performs 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 item recommendation. They compared their algorithm to the FolkRank algorithm, and found that tensor decomposition outperforms the latter. Wetzker et al. (2009) took a Probabilistic Latent Semantic Analysis (PLSA) approach, which assumes a latent lower dimensional topic model. They extended PLSA by estimating the topic model from both user-item occurrences as well as item-tag occurrences, and then linearly combined the output of the two models. They tested their approach on a large crawl of Delicious, and found that it significantly outperforms a popularity-based algorithm.

Content-based Recommendation Content-based filtering can be seen as a specialization of information filtering (Belkin and Croft, 1992). Content-based filtering has been applied to many different domains. Early work on content-based filtering included the NewsWeeder system by Lang (1995), which used the words contained in newsgroup messages as its features. 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. Mooney and Roy (2000) describe LIBRA, a content-based book recommender system. They crawled 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 new books for the user.

We are not the first to suggest the combination of CF with content-based filtering, as 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. CF is known, however, to have problems with users with unique tastes and so-called 'gray sheep' users with varied tastes that are hard to categorize (Burke, 2002). Content-based filtering methods are better at dealing with sparse, dynamic domains such as news filtering, and are better at recommending for non-average users. They are also known to produce more focused, high precision recommendations.

Several hybrid methods that try to combine the best of both worlds have been proposed over the years. 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 included metadata fields such as actors, directors, genre, titles, and tag lines. They used RIPPER, a rule-based machine learning algorithm to predict which items are interesting, and found that the combination of collaborative and content-based features produced the best results. 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.

4. Recommender Systems Fusion

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. In the second part of this article we will examine the possibilities of data fusion[11] in more detail. Instead of augmenting or combining features for recommendation, we will examine the effectiveness of combining the output of different recommendation runs.

This section is organized as follows. We start in Subsection 4.1 by discussing related work on data fusion in the fields of recommender systems, IR, and machine learning, and highlight some of the reasons why fusion is often successful. Then, in Subsection 4.2 we discuss the fusion techniques we will use, and in Subsection 4.3 we describe which individual runs we select for our fusion experiments. Subsection 4.4 describes the results of our fusion experiments, followed by an in-depth analysis in Subsection 4.5 of why fusion is successful for item recommendation.

4.1. Related Work

Fusing Recommender Systems In the past decade, the field of recommender systems has already seen several different approaches to combining different recommendation algorithms. Burke presented a taxonomy of seven different methods for creating hybrid recommendation algorithms (Burke, 2002). The *mixed* fusion method simply presents all outputs of the different individual algorithms. 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* algorithm switches between the different component algorithms based on certain criteria. For instance, in a start-up phase, where few ratings have been collected, initial recommendations could be based 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 (Claypool et al., 1999). In a *feature combination* approach, features from different types

[11] The term 'data fusion' can be ambiguous to a certain extent. In this article, 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 algorithms (i.e., collaborative information, content-based, or knowledge-based) are combined and used as the input feature set for a single recommendation algorithm. Burke (2002) 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, similar to learning-to-rank approaches in IR. In contrast, in a *feature augmented* hybrid algorithm one technique is employed to produce a rating of an item, after which only that rating is used as an input feature for the next recommendation technique. 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.

In this article we focus exclusively on the latter type of hybridization method: weighted combination. Relatively few weighted combination approaches have been described in the literature on recommender systems. 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.

Information Retrieval An important distinction to make here in the related work on data fusion in IR 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). Results fusion is similar to the weighted combination approach in recommender systems, and has received a great deal of attention in IR.

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 data 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.

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. 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, 1996). The validity of this principle has been confirmed in other studies for queries, documents, searchers, and retrieval algorithms (Skov et al., 2008; 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 meth-

ods 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 verified and extended by, among other, Lee (1997), Croft (2000), Renda and Straccia (2003), and Kamps and De Rijke (2004).

Why Does Fusion Work? 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 inversely related to what Vogt and Cottrell define as the *Skimming* effect: This 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 offered 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 missed by others. Clearly, one effect may counteract 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 *Skimming* effect).

4.2. Fusion Methods

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):

$$score_{norm} = \frac{score_{original} - score_{min}}{score_{max} - score_{min}}. \quad (1)$$

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 1.

We introduced six standard fusion methods in our discussion of the related work in Subsection 4.1. 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 $score_{norm}(i, r_n)$ in that run r_n. Let us also define the number of hits of an item in R, i.e., the number of runs that i occurs in, as $h(i,R) = |\{r \in R : i \in r\}|$. We can then represent all three combination methods CombSUM, CombMNZ, and CombANZ using the following equation:

$$score_{fused}(i) = h(i,R)^{\gamma} \cdot \sum_{n=1}^{N} w_n \cdot score_{norm}(i, r_n). \quad (2)$$

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. The w_n parameter allows us to assign different preference weights to each individual run in the range $[0,1]$. The CombSUM, CombMNZ, and CombANZ methods are all unweighted, which means that the preference weights for each run are equal. We set them to 1.0.

A alternative common fusion approach in both recommender systems and IR research is to do a *weighted* or *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 an optimal combination. We therefore also test weighted versions of the CombSUM, CombMNZ, and CombANZ combination methods and assign each run different preference weights. 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. When combining more than two runs, however, performing Even than 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. 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.

4.3. Selecting Runs for Fusion

To determine which runs we should fuse together, we adopt the intuition put 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: *data representations* and *algorithms*. For instance, we consider item-based filtering runs that use transaction patterns as a source of item similarity to be algorithmically identical to 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[12]. While many different combinations of individual runs are possible, we have selected only runs that differ on at least one of these dimensions. This means we consider three different combination types: (1) different representation, same algorithm, (2) same representation, different algorithm, and (3) different representation and different algorithm. We have likewise divided our experiments up into three groups corresponding to these three types of combinations.

Table 4 shows the fusion experiments we perform. In the first group of fusion experiments, A through D, we combine two runs with different algorithms but the same data representation. For instance, run A combines user-based CF and item-based CF, both with usage data similarity. In contrast, the next group of fusion experiments, E and F, combines three different runs that use

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

Table 4: 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. The fourth column shows how many runs are combined.

Run ID	Different representation	Different algorithm	Number of combined runs	Description
Fusion A	No	Yes	2	User-based CF and item-based CF with usage data similarity.
Fusion B	No	Yes	2	User-based CF and item-based CF with tag similarity.
Fusion C	No	Yes	2	Best content-based filtering runs.
Fusion D	No	Yes	2	Best hybrid filtering runs (user-based and item-based CF with metadata similarity).
Fusion E	Yes	No	3	User-based CF with usage data similarity, tag similarity, and metadata similarity.
Fusion F	Yes	No	3	Item-based CF with usage data similarity, tag similarity, and metadata similarity.
Fusion G	Yes	Yes	2	Best CF and metadata-based runs together.
Fusion H	Yes	Yes	4	All four CF runs combined.
Fusion I	Yes	Yes	4	All four metadata-based runs combined.
Fusion J	Yes	Yes	8	All eight individual combined.

the same algorithm, but different data representations. For instance, run E combines all three user-based CF runs with usage data, tag, and metadata similarity respectively. The third group of fusion experiments combine runs that use different algorithms as well as different representation of the data. Here, we also vary the number of runs combined. For instance, where fusion experiment G combines the best CF run with the best metadata-based run, fusion experiment J combines all eight individual runs.

4.4. Results

The results of the three groups of fusion experiments are listed in Tables 5–7. Table 5 lists the results of the fusion experiments where the data representation is kept constant, but different algorithms are combined. Table 6 shows the results of the experiments where different data representations were combined, while keeping the algorithm constant. Table 7 lists the outcomes of the experiments where both aspects were varied in the combination process.

Is Fusion Successful? What we see is that, overall, fusing recommendation results is successful: in 34 out of 40 fusion runs we find a performance increase over the best individual runs. If we consider the best-performing fusion runs for each data set, then we see improvements ranging from 12% to 72% in MAP scores. For instance, for CiteULike we see an improvement of 59% in MAP at 0.1556 using the weighted CombSUM method over the best individual run at 0.0978 in fusion experiment G.

Fusion Methods When we compare the results of the three different fusion methods, regardless of how they are weighted, we see that CombSUM and CombMNZ both consistently provide good performance. The difference between the two is never significant. In contrast, the CombANZ method performs poorly across the board.

When we compare the unweighted combination methods with 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 CiteULike. Here, the weighted CombSUM run achieves the data set-best MAP score 0.1506, which is significantly higher than the matching unweighted CombSUM run at 0.1180 ($p = 8.0 \cdot 10^{-6}$). 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. In terms of how the individual runs are weighted, we typically see that the best performing component runs are assigned the higher weights than the other runs. However, even poorly performing individual runs often receive non-zero weights, suggesting they still contribute to the final score in some way. We analyze the contributions of individual runs in more detail in Subsection 4.5.

Fusing Recommendation Aspects We considered two different recommendation aspects when deciding which runs to combine, representations and algo-

Table 5: Results of our fusion experiments that combine different algorithms while using the same data representations. 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 over all three sets of experiments in Tables 5 through 7. 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 Section 3 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	0.0322	0.0021	0.1273	0.0883$^{\triangledown}$
	CombMNZ	0.0320	0.0022	0.1273	0.0884$^{\triangledown}$
	CombANZ	0.0257	0.0010	0.0142$^{\triangledown}$	0.0112$^{\blacktriangledown}$
	Weighted CombSUM	0.0325	0.0022	0.1281	0.1005
	Weighted CombSUM	**0.0387**	0.0023	**0.1302**	**0.1008**
	Weighted CombSUM	0.0311	**0.0026**	0.1127	0.0371$^{\blacktriangledown}$
Fusion D	CombSUM	0.0311$^{\triangle}$	0.0051	0.1130	0.0856
	CombMNZ	0.0276$^{\triangle}$	0.0037	0.1143	0.0900
	CombANZ	0.0227	0.0036	0.0694$^{\triangledown}$	0.0412$^{\blacktriangledown}$
	Weighted CombSUM	**0.0312**$^{\triangle}$	**0.0051**	0.1461	0.0879$^{\blacktriangle}$
	Weighted CombMNZ	0.0275	0.0043	**0.1516**	**0.0947**$^{\blacktriangle}$
	Weighted CombANZ	0.0230	0.0042	0.1221	0.0539$^{\blacktriangledown}$
% Change over best individual run		+8.0%	+3.9%	+0.4%	+3.0%

Table 6: Results of our fusion experiments that combine different data representations while using the same algorithms. 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 over all three sets of experiments in Tables 5 through 7. 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 Section 3 is indicated in the bottom row.

Run	Method	bookmarks		articles	
		BibSonomy	Delicious	BibSonomy	CiteULike
Fusion E	CombSUM	**0.0289**	0.0051	0.0651	0.0728
	CombMNZ	0.0212	0.0041	0.0654	0.0720
	CombANZ	0.0286	0.0026	0.0258$^{\blacktriangledown}$	0.0647$^{\blacktriangledown}$
	Weighted CombSUM	0.0276	**0.0056**	0.0790	**0.0750**
	Weighted CombMNZ	0.0248	0.0048	**0.0879**	0.0747
	Weighted CombANZ	0.0263	0.0036	0.0626	0.0626$^{\blacktriangledown}$
Fusion F	CombSUM	0.0496	0.0038	**0.1575**	0.1437$^{\blacktriangle}$
	CombMNZ	0.0591$^{\triangle}$	0.0045	0.1392	**0.1449**$^{\blacktriangle}$
	CombANZ	0.0186$^{\triangledown}$	0.0039	0.0957	0.0810
	Weighted CombSUM	0.0409	**0.0113**	0.1185	0.1394$^{\blacktriangle}$
	Weighted CombMNZ	**0.0690**$^{\triangle}$	0.0108	0.1404	0.1419$^{\blacktriangle}$
	Weighted CombANZ	0.0180$^{\triangledown}$	0.0092	0.0694	0.0798
% Change over best individual run		+71.6%	+11.9%	+4.3%	+63.4%

Table 7: Results of our fusion experiments that combine different algorithms as well as different representations. 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 over all three sets of experiments in Tables 5 through 7. 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 Section 3 is indicated in the bottom row.

Run	Method	bookmarks		articles	
		BibSonomy	Delicious	BibSonomy	CiteULike
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	**0.0378**	0.0040	0.1245	0.1002
	CombMNZ	0.0365	0.0032	0.1214	0.1040
	CombANZ	0.0056$^{\triangledown}$	0.0010	0.0049$^{\triangledown}$	0.0073$^{\blacktriangledown}$
	Weighted CombSUM	0.0300	**0.0057**	0.1413	0.1112$^{\blacktriangle}$
	Weighted CombMNZ	0.0346	0.0045	**0.1518**	**0.1120**$^{\blacktriangle}$
	Weighted CombANZ	0.0114	0.0010	0.1245	0.0470$^{\blacktriangledown}$
Fusion J	CombSUM	0.0470	0.0055	0.1577$^{\triangle}$	0.1180
	CombMNZ	**0.0612**	0.0072	0.1571$^{\blacktriangle}$	0.1250$^{\triangle}$
	CombANZ	0.0083$^{\triangledown}$	0.0011$^{\blacktriangledown}$	0.0163	0.0157$^{\blacktriangledown}$
	Weighted CombSUM	0.0388	0.0083	0.1636$^{\blacktriangle}$	**0.1506**$^{\blacktriangle}$
	Weighted CombMNZ	0.0607	**0.0087**	**0.1853**$^{\blacktriangle}$	0.1401$^{\blacktriangle}$
	Weighted CombANZ	0.0169	0.0028$^{\blacktriangledown}$	0.0861	0.0489$^{\blacktriangledown}$
% Change over best individual run		+54.0%	+7.9%	+22.7%	+59.1%

rithms, and divided our fusion experiments up into three groups that combine runs based on differences in one of these aspects or both at the same time. While we see improvements of the fusion experiments in all three groups, there are differences between the groups.

If we look at the pairwise fusion runs A through D in Table 5, we can see that combining different algorithms that all use the same data representation yields only modest improvements over the best individual runs, with no increases in MAP exceeding 8%. In contrast, runs E and F in Table 6, where the algorithm is kept constant, but different data representations are combined, show much stronger improvements over the best individual runs, ranging from 4% to 72% increases in MAP. This suggests that of these two options it is often better to use the same algorithm yet combine different data sources, than to only vary the algorithm.

However, the results of fusion experiments G to J in Table 7 show that recommendation fusion where different recommendation aspects are combined, often performs better than when only one of the aspects is varied. Two of the best-performing fusion runs overall are runs where both aspects were varied. This resulted in MAP scores of 0.1853 and 0.1556 for Bibsonomy Articles and CiteULike respectively. The maximum improvement over the best individual runs range from 8% to 59%. While the runs from the second group also generate good recommendations, the overall picture suggests that varying both aspects is better across our two domains and four data sets.

Varying the Number of Runs A final question to ask is whether the number of combined runs influences the results. The answer is ambiguous. We do see that runs that fuse together four or eight individual runs tend to produce more accurate recommendations than pairwise fusion experiments. However, in some cases where the difference in MAP scores between the individual runs is too great, the inferior runs drag down the result below that of the best individual run. For example, fusion run J does not improve upon the best individual run on Delicious with a MAP of 0.0087 that is nearly 14% lower than the best individual run, item-based CF with tag similarity, at a MAP of 0.0101. This is probably caused by the significantly lower MAP scores of some of the individual runs on Delicious.

4.5. Fusion Analysis

While it is useful to determine which combination methods provide the best performance, we are also interested in finding out what it is 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 Subsection 4.1. 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 Bibsonomy Articles, where we combine all eight individual runs. 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 8 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, due to a better ranking of the items retrieved by run 2.

For identifying the enhancements due to increased

Table 8: 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	Org. MAP	Rel. docs	Prec.-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%
Fusion	0.1556	791	-		-	

Bibsonomy articles – Fusion run J – Weighted CombSUM

Run	Org. MAP	Rel. docs	Prec.-enh. MAP		Recall-enh. MAP	
Run 1	0.0865	82	0.1774	+105.1%	0.0860	-0.6%
Run 2	0.0737	90	0.1471	+99.6%	0.1008	+36.8%
Run 3	0.0459	46	0.0764	+66.4%	0.1535	+234.4%
Run 4	0.1100	54	0.1333	+21.2%	0.1599	+45.4%
Run 5	0.1279	108	0.1819	+42.2%	0.1304	+2.0%
Run 6	0.1190	105	0.1804	+51.6%	0.1228	+3.2%
Run 7	0.0155	37	0.0728	+369.7%	0.1259	+712.3%
Run 8	0.1510	89	0.1761	+16.6%	0.1588	+5.2%
Fusion	0.1853	118	-		-	

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 8. 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% over the original MAP score of 0.0887, entirely due to the improved recall of the fusion run.

The results in Table 8 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. The 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. These results suggest that there is stronger evidence for the *Chorus effect* than there is for the *Skimming effect*. The relative lack of recall-related improvements do not suggest the presence of a *Dark Horse effect*.

5. Conclusions

Recommending interesting content is an important recommendation task for social bookmarking websites. In this article we have examined different aspects of this item recommendation task.

Item Recommendation for Social Bookmarking We started by comparing eight different recommendation approaches that differ in algorithms and data representations used, to determine what work best for the task of item recommendation. We performed our experiments using a well-established evaluation metric on four data sets of different sizes, thereby addressing some of the problems of the previous work. We found that the best way of using the folksonomy for recommendation is to use an item-based CF algorithm that uses tagging data instead of the customary usage data. In addition, we found that recommendation algorithms based on metadata are competitive with algorithms that use the folksonomy as a source of recommendations. Among the four metadata-based approaches,

a profile-centric matching approach that matches user profiles directly against item profiles showed the most promise. However, there is no clear winner among the different approaches across data sets: the optimal approach is strongly dependent on the nature of the data set it is applied to.

Recommender Systems Fusion In addition to comparing individual recommendation approaches, we also investigated whether recommender systems fusion is a viable way of improving recommendation accuracy. We found that combining different recommendation runs indeed 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.

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. However, if a choice needs to be made, our results suggest that it is better to combine different data representations than it is to combine different algorithms.

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 smaller role in improving performance. Overall, we find strong evidence for the *Chorus effect* in our experiments.The lack of recall-related improvements suggests that we do not see the *Skimming effect* and *Dark Horse effect* occurring clearly in our fusion experiments.

5.1. Recommendations for Recommendation

Based on the results of our experiments, we can offer a set of recommendations for social bookmarking services seeking to implement a recommender system. 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. Recommendation using the information contained in the folksonomy is a good approach as well: here, we recommend implementing an item-based CF algorithm that uses tag overlap between items to calculate the similarity. 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.

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. However, we do believe that our findings are generalizable to these and other domains, such as movie recommendation, and other applications, such as e-commerce websites that offer a large catalog of items annotated with metadata and tagged by their users. An example of such an e-commerce website could be Amazon.com, where the majority of the items are annotated with rich metadata. We believe that in particular the lessons learned with regard to fusing recommendations could be transferred successfully to these other domains and applications.

5.2. Future Work

The work described in this article is an example of a *system-based* approach to evaluation. In such an approach, we try to simulate, as realistically as possible, the reaction of the user to different variants of the recommendation approaches in a controlled laboratory setting. A system-based evaluation, however, can only provide us with a provisional estimate of how well our individual and fused approaches are doing. User satisfaction is influenced by more than just recommendation accuracy (Herlocker et al., 2004). It would therefore be essential to follow up our work with a *user-based* evaluation on 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. This way we can determine with much greater certainty whether the approaches that perform best in system-based evaluation are also the preferred choice of the actual users of a system.

References

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 *Proceedings of SIGMOD '08*, pages 1323–1326, New York, NY, USA, 2008. ACM.

Javed A. Aslam and Mark Montague. Models for Metasearch. In *Proceedings of SIGIR '01*, pages 276–284, New York, NY, USA, 2001. ACM.

Shenghua Bao, Xiaoyuan Wu, Ben Fei, Guirong Xue, Zhong Su, and Yong Yu. Optimizing Web Search using Social Annotations. In *Proceedings of WWW '07*, 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 *Proceedings of SIGIR '94*, pages 173–181, New York, NY, 1994. Springer-Verlag New York, Inc.

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.

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 *Proceedings of SIGIR '93*, pages 339–346, New York, NY, USA, 1993. ACM.

Toine Bogers and Antal Van den Bosch. Collaborative and Content-based Filtering for Item Recommendation on Social Bookmarking Websites. In *Proceedings of the ACM RecSys '09 workshop on Recommender Systems and the Social Web*, pages 9–16, October 2009.

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.

Robin Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

Mark J. Carman, Marik Baillie, and Fabio Crestani. Tag Data and Personalized Information Retrieval. In *Proceedings of SSM '08*, pages 27–34, New York, NY, USA, 2008. ACM.

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, 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 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 *Proceedings of SIGIR '08*, 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.

Edward A. Fox and Joseph A. Shaw. Combination of Multiple Searches. In *TREC-2 Working Notes*, pages 243–252, 1994.

Scott A. Golder and Bernardo A. Huberman. Usage Patterns of Collaborative Tagging Systems. *Journal of Information Science*, 32 (2):198–208, 2006.

Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of SIGIR '99*, pages 230–237, New York, NY, USA, 1999. 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.

Paul Heymann, Daniel Ramage, and Hector Garcia-Molina. Social Tag Prediction. In *Proceedings of SIGIR '08*, pages 531–538, New York, NY, USA, July 2008. ACM.

Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information Retrieval in Folksonomies: Search and Ranking. In *Proceedings of ESWC '06*, 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.

Peter Ingwersen. Cognitive Perspectives of Information Retrieval Interaction: Elements of a Cognitive IR Theory. *Journal of Documentation*, 52(1):3–50, 1996.

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.

Robert Jäschke, Leandro Balby Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. Tag Recommendations in Folksonomies. In *Proceedings of PKDD 2007*, volume 4702 of *Lecture Notes in Computer Science*, pages 506–514. Springer Verlag, 2007.

Jaap Kamps and Maarten De Rijke. The Effectiveness of Combining Information Retrieval Strategies for European Languages. In *Proceedings of SAC '04*, pages 1073–1077, 2004.

Ken Lang. NewsWeeder: Learning to Filter Netnews. In *Proceedings of ICML '95*, 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.

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 *Proceedings of WWW '07*, pages 943–951, 2007.

Raymond J. Mooney and Loriene Roy. Content-Based Book Recommending Using Learning for Text Categorization. In *Proceedings of DL '00*, pages 195–204, New York, NY, 2000. ACM.

Reyn Nakamoto, Shinsuke Nakajima, Jun Miyazaki, and Shunsuke Uemura. Tag-Based Contextual Collaborative Filtering. In *Proceedings of the 18th IEICE Data Engineering Workshop*, 2007.

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.

Michael J. Pazzani. A Framework for Collaborative, Content-based and Demographic Filtering. *Artificial Intelligence Review*, 13(5): 393–408, 1999.

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.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of WWW '01*, pages 285–295, New York, NY, USA, 2001. 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 *Proceedings of SIGIR '08*, pages 515–522, New York, NY, USA, 2008. 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.

Panagiotis Symeonidis, Maria Ruxanda, Alexandros Nanopoulos, and Yannis Manolopoulos. Ternary Semantic Analysis of Social Tags for Personalized Music Recommendation. In *Proceedings of ISMIR '08*, pages 219–224, 2008.

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.

Karen H. L. Tso-Sutter, Leandro Balby Marinho, and Lars Schmidt-Thieme. Tag-aware Recommender Systems by Fusion of Collaborative Filtering Algorithms. In *Proceedings of SAC '08*, pages 1995–1999, New York, NY, USA, 2008. ACM.

Thomas Vander Wal. Explaining and Showing Broad and Narrow Folksonomies. Retrieved April 29, 2008, from http://www.personalinfocloud.com/2005/02/explaining_and_.html, 2005.

Christopher C. Vogt and Garrison W. Cottrell. Predicting the Performance of Linearly Combined IR Systems. In *Proceedings of SIGIR '98*, pages 190–196, New York, NY, 1998. ACM.

Ellen M. Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. Learning Collection Fusion Strategies. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of SIGIR '95*, pages 172–179, New York, NY, 1995. ACM.

Robert Wetzker, Winfried Umbrath, and Alan Said. A Hybrid Approach to Item Recommendation in Folksonomies. In *Proceedings of ESAIR '09*, pages 25–29, New York, NY, USA, 2009. ACM.

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.

Ding Zhou, Jiang Bian, Shuyi Zheng, Hongyuan Zha, and C. Lee Giles. Exploring Social Annotations for Information Retrieval. In *Proceedings of WWW '08*, pages 715–724, New York, NY, USA, 2008. ACM.