Collaborative and Content-based Filtering for Item Recommendation on Social Bookmarking Websites

Toine Bogers
ILK / Tilburg centre for Creative Computing
Tilburg University
P.O. Box 90153, 5000 LE
Tilburg, The Netherlands
A.M.Bogers@uvt.nl

Antal van den Bosch
ILK / Tilburg centre for Creative Computing
Tilburg University
P.O. Box 90153, 5000 LE
Tilburg, The Netherlands
Antal.vdnBosch@uvt.nl

ABSTRACT

Social bookmarking websites allow users to store, organize, and search bookmarks of web pages. Users of these services can annotate their bookmarks by using informal tags and other metadata, such as titles, descriptions, etc. In this paper, we focus on the task of item recommendation for social bookmarking websites, i.e. predicting which unseen bookmarks a user might like based on his or her profile. We examine how we can incorporate the tags and other metadata into a nearest-neighbor collaborative filtering (CF) algorithm, by replacing the traditional usage-based similarity metrics by tag overlap, and by fusing tag-based similarity with usage-based similarity. In addition, we perform experiments with content-based filtering by using the metadata content to recommend interesting items. We generate recommendations directly based on Kullback-Leibler divergence of the metadata language models, and we explore the use of this metadata in calculating user and item similarities. We perform our experiments on three data sets from two different domains: Delicious, CiteULike and BibSonomy.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software

General Terms

Algorithms, Measurement, Performance, Experimentation

Keywords

Recommender systems, social bookmarking, folksonomies, collaborative filtering, content-based filtering

1. INTRODUCTION

Recommender systems belong to a class of personalized 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. *Social bookmarking websites*, with their emphasis on open collaborative information access, offer an ideal scenario for the application of recommender systems technology. They allow users to manage their favorite bookmarks online through a web interface and, in many cases, allow their users to tag the content they added to the system with keywords. The aggregation of these tags, the *folksonomy*, is an extra annotation layer connecting users and items. The underlying application then makes all information sharable among users. The success of such websites partly depends on how the connections between users, items, and tags are exploited.

Social bookmarking websites also offer possibilities for attaching item-specific metadata to each item, such as the item's title or summary. This additional metadata could also be used to support the recommendation process. Using item metadata to boost performance is not new in recommender systems research, but typically such content-related information is attached to the items, and is therefore the same across all users [23]. On the other hand, the tags assigned to items are specific to the user who added them. We know of no approaches to item recommendation for social bookmarking that investigate the use of such metadata.

In this paper we focus on the question how we can make use the information represented by the folksonomy and the item metadata to boost the performance of traditional collaborative filtering algorithms. We make the following contributions. First, we examine different ways of extending the standard nearest-neighbor algorithm with information about tagging behavior. Second, we explore how best to use the metadata for recommendation by proposing four different algorithms. Finally, we perform our experiments on publicly available data sets with standardized evaluation metrics to promote verifiability. The remainder of this paper is structured as follows. We start by introducing our data sets and experimental setup in the next section. In Section 3 we describe and compare different CF algorithms that operate on the folksonomy to generate recommendations. Section 4 we describe how we exploit the metadata in our data sets to generate item recommendations. We describe the related work in Section 5 and conclude in Section 6.

2. METHODOLOGY

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

ACM RecSys '09 Workshop on Recommender Systems and the Social Web
October 25, 2009, New York, NY, USA.

BibSonomy) and the other two cover the domain of scientific articles (Delicious and BibSonomy).

CiteULike is a social bookmarking service that allows its users to add their academic reference library to an online profile[1]. Articles can be stored with their metadata, abstracts, and links to the papers at the publishers' sites. Users can also add personal comments and tags. CiteULike offers 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.

BibSonomy is a social bookmarking service for sharing Web page bookmarks and reference lists of scientific articles[2]. Items are stored and represented by their BibTeX metadata representations. These can include 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 [13]. 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[3]. The organizers made 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).

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[4]. 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 their entire profiles.

2.1.1 *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 [11]. 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. In addition, we filter out all items that occur only once, as well as all untagged posts. We were able to identify and filter out most of the spam content in the CiteULike and BibSonomy data sets. We refer the reader to [5] for more details about this process. Table 1 lists the statistics of our four data sets after filtering.

[1] http://www.citeulike.org/
[2] http://www.bibsonomy.org/
[3] http://www.kde.cs.uni-kassel.de/ws/rsdc08/
[4] http://www.delicious.com/

2.2 Experimental Setup & Evaluation

In order to evaluate and compare different recommender algorithms, we need a proper framework for experimentation and evaluation. Recommender systems evaluation—and the differences with IR evaluation—has been addressed by, among others, [11]. We evaluate the "Find Good Items" task, also known as Top-N recommendation, where users are provided with a ranked list of recommended items based on their personal profile. 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 do 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 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 macro-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. Herlocker et al. [11] 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 over each relevant retrieved item. For determining 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 $^{\vartriangle}$ (and $^{\triangledown}$) for $\alpha = .05$ and $^{\blacktriangle}$ (and $^{\blacktriangledown}$) for $\alpha = .01$.

3. FOLKSONOMIC RECOMMENDATION

We start by establishing some notation and definitions of the task at hand, based in part on notation by [9]. In the social bookmarking setting, users post items to their personal profiles and can choose to label them with one or more tags. We define a folksonomy to be the tripartite graph that emerges from this collaborative annotation of items. The resulting ternary relations that make up the tripartite graph can be represented as a 3D matrix of users, items, and tags. Figure 1 illustrates this view. We refer to the 3D matrix as $\mathbf{D}(u_k, i_l, t_m)$. Here, each element $d(k, l, m)$ of this matrix indicates if user u_k (with $k = \{1, \ldots, K\}$) tagged item i_l (with $l = \{1, \ldots, L\}$) with tag t_m (with $m = \{1, \ldots, M\}$), where a value of 1 indicates the ternary relation is present in the folksonomy.

In conventional recommender systems, the user-item matrix contains ratings information. These ratings can be *explicit*, when they are entered directly by the user, or *implicit*, when they are inferred from user behavior. In our case we have implicit, unary ratings where all items that were added by a user receive a rating of 1. We extract this ratings matrix $\mathbf{R}(u_k, i_l)$ for all user-item pairs directly from the tripartite graph. We denote its individual elements by $x_{k,l} = \{1, \emptyset\}$. Each user is represented in this matrix as its user profile row vector $\mathbf{u}_k$, which lists the items that user added to his or her profile. Items are represented by the column vectors of $\mathbf{R}$ which represent the item profile vectors $\mathbf{i}_l$ that contain all users that have added that item. As shown in Figure 1, we can also extract a user-item matrix from $\mathbf{D}$ by aggregating over the tag dimension. We then

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

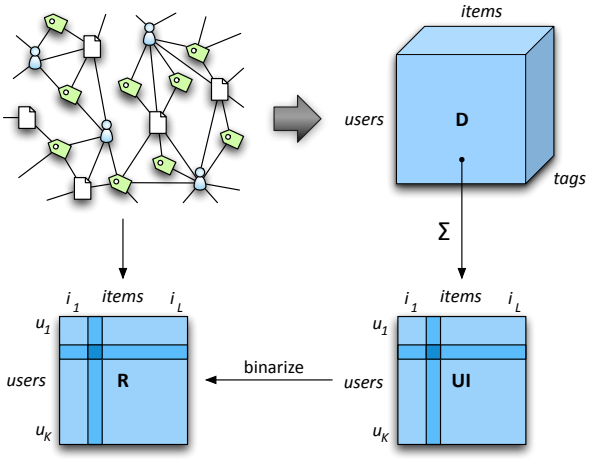

Figure 1: Representing the folksonomy graph as a 3D matrix. The ratings matrix R is derived from the tripartite graph itself, and directly represents what items were added by which users. Aggregation over the tag dimension of D gives us matrix UI, containing the tag counts for each user-item pair. We can obtain R by binarizing the values in UI..

obtain the $K \times L$ user-item matrix $\mathbf{UI}(u_k, i_l) = \sum_{m=1}^{M} \mathbf{D}(u_k, i_l, t_m)$, specifying how many tags each user assigned to each item. Because we filtered our data sets to include only tagged content, our ratings matrix $\mathbf{R}$ is the same as a binary version of $\mathbf{UI}$. Similar to the way we defined $\mathbf{UI}$ we can also aggregate the content of $\mathbf{D}$ over the user and the item dimensions. We define the $K \times M$ user-tag matrix $\mathbf{UT}(u_k, t_m) = \sum_{l=1}^{L} \mathbf{D}(u_k, i_l, t_m)$, specifying how often each user used certain tag to annotate his or her items. Individual elements of $\mathbf{UT}$ are denoted by $y_{k,m}$. We define the $L \times M$ item-tag matrix $\mathbf{IT}(i_l, t_m) = \sum_{k=1}^{K} \mathbf{D}(u_k, i_l, t_m)$, indicating how many users assigned a certain tag to an item. Individual elements of $\mathbf{IT}$ are denoted by $z_{l,m}$. We also define binary versions of $\mathbf{UT}$ and $\mathbf{IT}$ as $\mathbf{UT}_{binary}$ and $\mathbf{IT}_{binary}$. The row vectors of the $\mathbf{UT}$ and $\mathbf{IT}$ matrices represent the user tag profiles $\mathbf{d}_k$ and item tag profiles $\mathbf{f}_l$ respectively. They list what tags have been assigned by a user to his items, or to an item by its users. Formally, the goal of each of the recommendation algorithms discussed in this paper is to produce a ranking of all items l that are not yet in the profile of the active user u_k (i.e., $x_{k,l} = \emptyset$). To this end, we predict a score $\widehat{x}_{k,l}$ for each item that represents the likelihood of that item being relevant for the active user. The final recommendations for a user are generated by ranking all items i_l by their predicted score $\widehat{x}_{k,l}$.

3.1 Baseline Recommendation Algorithms

A common and well-understood source of information for recommendation is usage patterns of adding and rating items. The class of algorithms that exploit such patterns for recommendation purposes are called Collaborative Filtering algorithms (CF). In this paper we focus on using and extending the k-Nearest Neighbor (k-NN) algorithm. We pick the k-NN algorithm because it is a well understood algorithm that can intuitively be extended to include other information in addition to transaction patterns [7, 10]. There are two flavors of the k-NN algorithm for CF: *user-based filtering* and *item-based filtering*. In user-based filtering, we locate the users most similar to the active users, and look among their items for new recommendations. In item-based filtering, we locate the most similar items for each of the active user's items and aggregate these into a list of predicted items.

User-based Filtering.

In the first step of user-based filtering we calculate the similarities between pairs of users to identify the most similar users for an active user. Many different similarity metrics have been proposed and evaluated over time, such as Pearson's correlation coefficient and cosine similarity [6]. We use the cosine similarity in our experiments as it has often been used successfully on data sets with implicit ratings [6, 19]. We calculate the cosine similarity between the active user u_k and another user u_a on the user profile vectors $\mathbf{u}_k$ and $\mathbf{u}_a$ as $sim_{cosine}(u_k, u_a) = \frac{\mathbf{u}_k \cdot \mathbf{u}_a}{\|\mathbf{u}_k\| \, \|\mathbf{u}_a\|}$.

The next step in user-based filtering is to determine the top N similar users (or items) for user u_k. We denote this set as the Set of Similar Users $SSU(u_k)$, which are the top N users of the set of all users u_a, ranked by their cosine similarity. For each user u_a, we only consider those items that u_a added to his profile ($x_{a,l} \neq \emptyset$). Using this set of nearest neighbors we generate the final prediction scores $\widehat{x}_{k,l}$ for each unseen item i_l as $\widehat{x}_{k,l} = \sum_{u_a \in SSU(u_k)} sim_{cosine}(u_k, u_a)$. Here, 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 added item i_l (i.e. $x_{a,l} \neq \emptyset$).

A recurring observation from the literature about CF algorithms is that universally liked items are not as useful for capturing the similarity between users as less common items, see e.g. [6]. We therefore perform two runs: the 'vanilla' base run described above (u-bin-sim) and a run where the values in the user vectors are weighted by the *inverse user frequencies* of the items (u-bin-idf-sim).

Item-based Filtering.

The item-based k-NN algorithm operates analogously to the user-based filtering algorithm [19]. Instead of comparing users directly,

Table 2: Results of the folksonomic recommendation runs. Reported are the MAP scores as well as the optimal number of neighbors N. Best-performing runs for each data group of approaches are printed in bold. Best-performing tag overlap runs for both user-based and item-based are printed in bold. The percentage difference between the best baseline CF runs and the best tag overlap runs are indicated after each type.

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

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 test items of the active user u_k and the other items that user has not yet added (so $x_{k,l} = \emptyset$). Similarity between two items i_l and i_b is calculated on the item profile vectors $\mathbf{i}_l$ and $\mathbf{i}_b$ as $sim_{cosine}(i_l, i_b) = \frac{\mathbf{i}_l \cdot \mathbf{i}_b}{||\mathbf{i}_l|| \, ||\mathbf{i}_b||}$. Next, we identify the top N most similar items for each of the active user's items i_l separately. We define this neighborhood as the Set of Similar Items $SSI(i_l)$, where we select the top N of all items not already in the active user's profile, ranked on their cosine similarity $sim_{cosine}(i_l, i_b)$ to item i_l. Using this set of nearest neighbors we generate the final prediction score $\widehat{x}_{k,l}$ for each unseen item i_l as $\widehat{x}_{k,l} = \sum_{i_b \in SSI(i_l)} sim_{cosine}(i_l, i_b)$. Here, the predicted score is the sum of the similarity values (between 0 and 1) of all the most similar items that were added by user u_k (i.e. $x_{k,b} \neq \emptyset$). Analogous to user-based filtering, we can also suppress the influence of the most 'popular' users, i.e. users that have added a disproportionately large number of items to their profile, such as bots or spam users. We refer to the item-based filtering runs weighted with the *inverse item frequency* as i-bin-idf-sim and to the unweighted runs as i-bin-sim.

3.2 Tag Overlap Similarity

The folksonomies present in our four data sets each constitute an extra layer of connections between user and items. We exploit this extra layer for determining another type of similarity between users or items. For instance, users that assign many of the same tags can be seen as similar, and items that are often assigned the same tags can also be seen as similar.

We restrict ourselves to comparing three common similarity metrics: Jaccard overlap, Dice's coefficient, and the cosine similarity. We use these similarities as the basis for item recommendation. The only difference between this approach and the standard CF algorithm is in the first step, where the similarities are calculated. For user-based filtering, we calculate tag overlap on the **UT** matrix or on the binarized version $\mathbf{UT}_{binary}$, depending on the metric. Both the Jaccard overlap and Dice's coefficient are set-based metrics, which means we calculate them on the binary vectors from the $\mathbf{UT}_{binary}$ matrix. The Jaccard Overlap $sim_{Jaccard}(d_k, d_a)$ between two users d_k and d_a is defined as $\frac{|\mathbf{d}_k \cap \mathbf{d}_a|}{|\mathbf{d}_k \cup \mathbf{d}_a|}$. Dice's coefficient $sim_{Dice}(d_k, d_a)$ is defined as $\frac{2|\mathbf{d}_k \cap \mathbf{d}_a|}{|\mathbf{d}_k| + |\mathbf{d}_a|}$. We refer to the user-based runs with the Jaccard overlap and Dice's coefficient as ut-jaccard-sim and ut-dice-sim respectively. The cosine similarity is calculated in three different ways. First, we calculate it on the regular tag count vectors $\mathbf{d}_k$ and $\mathbf{d}_a$ from **UT** as ut-tf-sim, and on the binary vectors from the $\mathbf{UT}_{binary}$ matrix as ut-bin-sim. In addition, we also experiment with *idf*-weighting of the tags in the user tag count vectors as we did before. We refer to this run as ut-tfidf-sim. The item-based versions of these similarity metrics are calculated on the **IT** and $\mathbf{IT}_{binary}$ matrices. We refer to these five item-based runs as it-jaccard-sim, it-dice-sim, it-bin-sim, it-tf-sim, and it-tfidf-sim.

3.3 Results & Discussion

Table 2 compares the results of our baseline CF runs that employ usage-based similarities to the runs that use overlap in tagging behavior as a source of user and item similarity. We see that the user-based filtering baseline outperforms item-based filtering on three of four data sets; only on CiteULike does item-based filtering work better, where this difference is also statistically significant ($p < 0.05$). The other differences between user-based and item-based filtering are not significant. There appears to be no clear or statistically significant advantage to applying *idf*-weighting to the profile vectors. An explanation for the advantage of user-based filtering is that, according to Table 1, the average number of items per user is much higher than the average number of users per item. Calculating a meaningful overlap between user profile vectors could therefore be more robust than between item profile vectors.

As for the results with tag overlap, we observe that item similarities based on tag overlap work well for item-based filtering, as three of our four data sets show considerable improvements over the best CF baseline runs. Performance increases range from 27% on BibSonomy Articles to almost 120% 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 for almost all variants on all data sets, with performance decreases ranging from 40% to 63%. This means that using tag overlap in item-based filtering 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 added the item. This means that, on average, item profile vectors from the **IT** matrix are less sparse than item profile vectors from the **UI** matrix, 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.

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. The results of the different tag overlap metrics tend to be close together and differences between them are not statistically significant. Even though the best-performing similarity metrics are dependent on the data set, we do see that the metrics operating on the binary vectors from the $\mathbf{UT}_{binary}$ and $\mathbf{IT}_{binary}$ matrices are consistently among the top performers.

In general, it appears that bookmark recommendation is more difficult than article recommendation. We believe this is due to a difference 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 more different 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: 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.

4. RECOMMENDATION USING METADATA

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. In this section we investigate the role such metadata can play in recommending interesting bookmarks or references. We propose two different approaches: *content-based filtering* and *hybrid filtering*. Before we move on to describing these in Sections 4.1 and 4.2, we first take a closer look at the metadata we have available.

In our approach 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`. The intuition behind assigning metadata fields to the item-intrinsic category is that these fields can be used as standalone 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 `MONTH` or `PAGES`—cannot be used to directly generate appropriate recommendations. We performed experimental runs using the metadata of each of our intrinsic fields separately. In addition, we experimented with the combination of all intrinsic fields, and with runs that combined all intrinsic and extrinsic fields, resulting in a total of 34 runs per algorithm. We did not test the extrinsic fields separately. Due to space restrictions we only report the results of the best runs for each algorithm.

4.1 Content-based Filtering

The first approach we propose is content-based filtering where 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 use this profile to rank-order the unseen items by similarity to the profile, thereby approximating possible interest in those items. Figure 2 illustrates two different algorithms we propose for content-based filtering: *profile-centric matching* and *post-centric matching*.

The difference between our two content-based filtering algorithms is the level of aggregation. In our profile-centric matching approach, we collate all of a user's assigned metadata into a single user profile. The intuition here is that by aggregating all of the metadata assigned by a user we can completely capture his or her interests. Similarly, we construct item profiles that collate all of the metadata assigned to those items by all users in the training set. We then match the active user profiles against the item profiles on similarity to produce a ranking of all items, as illustrated in the top half of Figure 2. After removing the items already in the active user's profile, we are left with the final rank-ordered list of recommendations.

In contrast, 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, as illustrated in the bottom half of Figure 2. 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 $score(i_l)$.

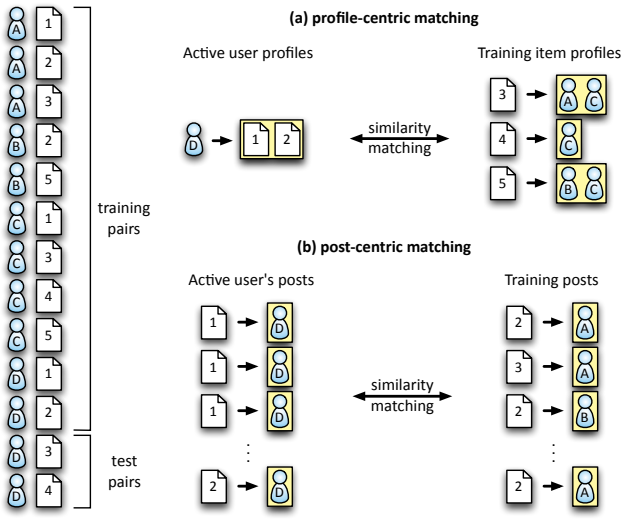

Figure 2: Visualization of our two content-based filtering approaches to item recommendation for a small toy data set.

In both content-based filtering algorithms, we approach the recommendation process from an IR perspective and restrict ourselves to measuring textual similarity. We use the open-source retrieval toolkit Lemur to calculate the similarities between the different user and item profiles. The Lemur toolkit[5] implements different retrieval methods based on language modeling [20]. 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 retrieval 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.

4.2 Hybrid Filtering

In addition to focusing solely on using the metadata for recommendation, we also consider a hybrid approach that joins content-based filtering and CF, in the hope of combining the best of both worlds. Many different combination methods have been proposed in earlier work [7]. 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 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 the previous section. The user and item similarities we derive in this way are then plugged into the standard memory-based CF algorithms as described in Section 3.1. The resulting algorithm is a feature-augmented hybrid of CF and content-based filtering.

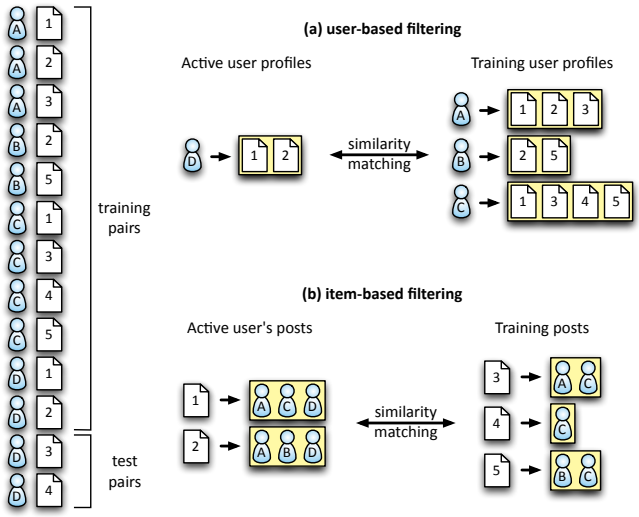

Figure 3: Visualization of our two hybrid filtering approaches to item recommendation for a small toy data set.

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 4.1, 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. Again, we used the open-source retrieval toolkit Lemur to calculate the similarities between the different metadata representations, with the same experimental settings as described in Section 4.1.

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

4.3 Results & Discussion

Table 3 contains the best runs for each of the four metadata-based algorithms, as well as our best CF run from Section 3. What we see, is that 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. All of our best metadata runs use the combined metadata fields. On their own, each field can be seen as an imperfect representation of the items and users, but combined they alleviate each others weak points and better represent the content than they do separately. Only on the Delicious data set do all metadata-based approaches perform significantly worse than the CF runs. Unfortunately, we do not have an explanation for this. When we compare the metadata-based approaches with each other, we see that most differences are not statistically significant. On the Bibsonomy Articles data set, the item-centric hybrid filtering approach is significantly better than the user-centric approach ($p < 0.05$). On the CiteULike data set, the profile-centric approach also significantly outperforms the post-centric and user-centric approaches.

In general, we 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 117% ($p < 10^{-6}$). Only on the Delicious data set does post-centric matching perform significantly better ($p < 0.05$). This advantage of the profile-centric approach is strongest on the article data sets where the profile-centric approach performs best for 75% of the all runs with different fields. In the case of hybrid filtering, the item-centric approach outperforms the user-centric approach on three of our four data sets. On the CiteULike and Bibsonomy Articles data sets these differences are statistically significant and especially large at 268% ($p < 0.05$) and 112% respectively ($p < 0.01$).

While we do not have room to report the results of all individual intrinsic field runs, we can report on our general findings. For all four approaches, the best-performing single fields are AUTHOR, DESCRIPTION, TAGS, and TITLE, which provide the best individual results on all four data sets for all approaches. This is not surprising, as these fields are the least sparsely filled of all the intrinsic fields. In addition, these four fields are also aimed directly at describing the content of the items, more so than the conference or journal titles or the editors. Another interesting observation is that the TITLE field served as a better source of user and item similarity on the article data sets than on the bookmark data sets. This is because titles assigned to bookmarks are more variable than titles assigned to scientific articles, leading to this performance gap.

Table 3: Results comparison of the best metadata-based runs with our best folksonomic CF runs. Reported are the MAP scores as well as the optimal number of neighbors N where applicable. The best-performing runs are printed in bold. The percentage difference between our best meta-data approaches and the best CF runs is listed in the bottom row.

Runs	bookmarks				articles			
	BibSonomy		Delicious		BibSonomy		CiteULike	
	MAP	N	MAP	N	MAP	N	MAP	N
Best CF run	0.0370	3	**0.0101**	23	0.1100	7	0.0887	30
	(if-jaccard-sim)		(if-bin-sim)		(if-tfidf-sim)		(i-bin-idf-sim)	
Profile-centric filtering	**0.0402**	-	0.0014$^{\blacktriangledown}$	-	0.1279	-	**0.0987**	-
	(all intrinsic)		(TITLE)		(all intrinsic)		(all extrinsic)	
Post-centric filtering	0.0259	-	0.0036$^{\triangledown}$	-	0.1190	-	0.0455$^{\blacktriangledown}$	-
	(all intrinsic)		(TAGS)		(all intrinsic)		(all extrinsic)	
User-centric hybrid filtering	0.0218	2	0.0039$^{\triangledown}$	13	0.0410	2	0.0608$^{\blacktriangledown}$	2
	(URL)		(all intrinsic)		(TITLE)		(TITLE)	
Item-centric hybrid filtering	0.0399	11	0.0017$^{\blacktriangledown}$	8	**0.1510**	21	0.0746	21
	(TAGS)		(all intrinsic)		(all intrinsic)		(TAGS)	
% Change over best CF run	+8.6%		-61.3%		+37.2%		+11.3%	

5. RELATED WORK

5.1 Folksonomic Recommendation

One of the first approaches to recommendation for social bookmarking websites was presented by [12], 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 [17] that uses the steady state node probabilities as the basis for ranking their recommendations. Clements et al. [9] 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. Approaches that resemble our use of tag overlap for calculating similarities between users and items include [2], [16], and [22]. Tso-Sutter et al. [23] 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. [21] 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 [12], and found that tensor decomposition outperforms the latter. Wetzker et al. [24] 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.

5.2 Exploiting Metadata for Recommendation

While a significant amount of research has focused on Collaborative Filtering for recommending interesting items, there has also been considerable work on content-based filtering, which can be seen as an extension of the work done on information filtering. Content-based filtering has been applied to many different domains. Early work on content-based filtering included the NewsWeeder system by Lang et al. [14], which used the words contained in newsgroup messages as its features. Alspector et al. [1] 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 et al. [15] describe Libra, a content-based book recommender system. They crawled the book metadata from the Amazon website and represented each book as a bag-of-words vector. They then used a Naive Bayes classifier to learn user profiles and to rank-order unseen books for the user.

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. 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. Basu et al. [3] 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. [8] 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 received 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. Baudisch [4] proposed an innovative approach to incorporating metadata into CF algorithms by joining the metadata descriptions to the user-item matrix as additional users.

6. CONCLUSIONS

In this paper we have presented a range of collaborative and content-based approaches to item recommendation on social bookmarking websites. Our algorithms were evaluated on four realistic data sets of different domains, and compared to two external, state-of-the-art approaches. Let us step back now and take stock of our findings. Tags represent an additional layer of information in the folksonomy that binds users and items together. These tags can be used successfully to improve the recommendations of standard nearest-neighbor algorithms, but this depends on the algorithm. For item-based filtering, using tags for calculating item similarity alleviates sparsity and results in better performance. At the user level, however, tags do not offer the same benefits.

Metadata can also be used successfully to generate item recommendations for social bookmarking websites. 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.

For future work, we intend to examine the benefits of data fusion. The tag-aware fusion approach by Tso-Sutter et al. [23] demonstrates the potential of fusing together the outputs of different recommendations algorithms and representations.

Acknowledgments

The work described in this paper was funded by SenterNovem / the Dutch Ministry of Economics Affairs as part of the IOP-MMI À Propos project, and by the Netherlands Organization for Scientific Research as part of the NWO Vernieuwingsimpuls program.

7. REFERENCES

[1] J. Alspector, A. Koicz, and N. 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.

[2] S. Amer-Yahia, A. Galland, J. Stoyanovich, and C. Yu. From del.icio.us to x.qui.site: Recommendations in Social Tagging Sites. In *Proceedings of SIGMOD '08*, pp. 1323–1326, New York, NY, USA, 2008. ACM.

[3] C. Basu, H. Hirsh, and W. W. Cohen. Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pp. 714–720, 1998.

[4] P. Baudisch. Joining Collaborative and Content-based Filtering. In *Proceedings of the ACM CHI Workshop on Interacting with Recommender Systems*. ACM Press, May 1999.

[5] T. Bogers and A. Van den Bosch. Using Language Modeling for Spam Detection in Social Reference Manager Websites. In R. Aly, C. Hauff, I. den Hamer, D. Hiemstra, T. Huibers, and F. de Jong, editors, *Proceedings of the 9th Belgian-Dutch Information Retrieval Workshop (DIR 2009)*, pp. 87–94, Enschede, February 2009.

[6] J. S. Breese, D. Heckerman, and C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 43–52, 1998.

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

[8] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining Content-Based and Collaborative Filters in an Online Newspaper. In *Proceedings of ACM SIGIR Workshop on Recommender Systems*, August 1999.

[9] M. Clements, A. P. de Vries, and M. J. Reinders. Optimizing Single Term Queries using a Personalized Markov Random Walk over the Social Graph. In *Proceedings of ESAIR '08*, 2008.

[10] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of SIGIR '99:*, pp. 230–237, New York, NY, USA, 1999. ACM.

[11] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[12] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information Retrieval in Folksonomies: Search and Ranking. In *Proceedings of ESWC '06*, 2006.

[13] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. BibSonomy: A Social Bookmark and Publication Sharing System. In *Proceedings of the Conceptual Structures Tool Interoperability Workshop at ICCS 2006*, pp. 87–102, 2006.

[14] K. Lang. NewsWeeder: Learning to Filter Netnews. In *Proceedings of ICML '95*, pp. 331–339, San Mateo, CA, USA, 1995. Morgan Kaufmann.

[15] R. J. Mooney and L. Roy. Content-Based Book Recommending Using Learning for Text Categorization. In *Proceedings of DL '00*, pp. 195–204, New York, NY, 2000. ACM Press.

[16] R. Nakamoto, S. Nakajima, J. Miyazaki, and S. Uemura. Tag-Based Contextual Collaborative Filtering. In *Proceedings of the 18th IEICE Data Engineering Workshop*, 2007.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[18] G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[19] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of WWW '01*, pp. 285–295, New York, NY, USA, 2001. ACM.

[20] T. Strohman, D. Metzler, and W. B. Croft. Indri: A Language Model-based Search Engine for Complex Queries. In *Proceedings of ICIA '05*, May 2005.

[21] P. Symeonidis, M. Ruxanda, A. Nanopoulos, and Y. Manolopoulos. Ternary Semantic Analysis of Social Tags for Personalized Music Recommendation. In *Proceedings of ISMIR '08*, pp. 219–224, 2008.

[22] M. Szomszor, C. Cattuto, H. Alani, K. O'Hara, A. Baldassarri, V. Loreto, and V. D. 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.

[23] K. H. L. Tso-Sutter, L. B. Marinho, and L. Schmidt-Thieme. Tag-aware Recommender Systems by Fusion of Collaborative Filtering Algorithms. In *Proceedings of SAC '08*, pp. 1995–1999, New York, NY, 2008. ACM.

[24] R. Wetzker, W. Umbrath, and A. Said. A Hybrid Approach to Item Recommendation in Folksonomies. In *Proceedings of ESAIR '09*, pages 25–29, New York, NY, USA, 2009. ACM.