# Recommending Scientific Articles Using CiteULike

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

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

## ABSTRACT

We describe the use of the social reference management website CiteULike for recommending scientific articles to users, based on their reference library. We test three different collaborative filtering algorithms, and find that user-based filtering performs best. A temporal analysis of the data indexed by CiteULike shows that it takes about two years for the cold-start problem to disappear and recommendation performance to improve.

## Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.4 Systems and Software; H.3.5 Online Information Services; H.3.7 Digital Libraries

## General Terms

Algorithms, Measurement, Performance, Experimentation

## 1. INTRODUCTION

One of the trends within the Web 2.0 paradigm is a shift in information access from local and solitary, to global and collaborative. Instead of storing, managing, and accessing personal information on only one specific computer or browser, personal information management and access has been moving more and more to the Web. Social bookmarking websites are clear cases in point: instead of keeping a local copy of pointers to favorite URLs, users can instead store and access their bookmarks online through a Web interface. The underlying application then makes all stored information sharable among users, allowing for improved searching and generating recommendations between users with similar interests.

A special kind of social bookmarking services—and the focus of our paper—are social reference managers such as CiteULike, Connotea, Bibsonomy, and 2Collab[1] that aid users in managing their reference collection. All of these services allow users to bookmark any Web page or reference they choose, and in addition they offer special functionality for certain academic resources, such as linking to online versions of papers and special access to metadata specific to academic resources.

All four mentioned bibliographical reference managers encourage users to organize their references with one or more tags, or keywords. These in turn enable users to view all references, from any user, associated with a chosen tag, as well as information about the popularity of a reference. This same linking is also applied to the author level, so that users can browse other users who added references to publications written by a specific author. These features can help users to better cope with the information overload that is as overwhelming in the academic community as it is on the Web, with an ever-increasing number of journals, books, and conference proceedings being published every year. This overload makes it hard to keep up with interesting new work, or to get a complete overview of relevant literature on specific topics.

For these features to be effective, active use of the online system on the part of the user (searching, browsing) is needed. Our interest lies in using recommender systems to relieve this burden and automatically find interesting and related reading material for the user. A recommender system is a type of personalized information filtering technology used to identify a sets of items that are likely to be of interest to a certain user. One particular class of recommendation algorithms is collaborative filtering (CF), that base recommendations on the opinions or actions of other like-minded users. The motivation here is that a user will be more satisfied with recommended items that are liked by like-minded users, than by items that are picked randomly or based on overall popularity.

In this paper, we focus on using one of these social reference managers, CiteULike, to generate reading lists for scientific articles based on a user's online reference library. We describe the construction of a test collection based on the services offered by CiteULike and apply three different CF algorithms to our data. We also analyze the data across its temporal dimension: we use publicly available activity logs to determine how recommendation performance changes as the website grows over time.

The paper is structured as follows. We discuss related work in Section 2. We discuss CiteULike, how our test collection was created, and what issues we ran into in greater detail in Section 3. In the following Section 4 we describe our experimental setup and evaluation, followed by the results in Section 5. Section 6 contains the results of our temporal analysis of the different algorithms. We conclude in Section 7 and highlight possible future work.

## 2. RELATED WORK

Most of the work related to recommending interesting information with respect to the user's current interest or task has focused on creating information management agents. Maes (1997) was among the first to signal the need for information filtering agents that can

[1] Available at http://www.citeulike.org, http://www.connotea.org, http://www.bibsonomy.org, and http://www.2collab.com respectively.


*RecSys'08,* October 23–25, 2008, Lausanne, Switzerland.

reduce overload [8]. Since then, several types of agents have been prototyped and developed for many different fields, such as the Web, music, and academic writing. See Montaner et al. (2003) for a comprehensive overview of agents available on the Web.

Yet there have been only a handful of approaches to recommending interesting academic articles to users, McNee et al. (2006) arguably being the most prominent one. McNee frames the interaction between users and recommender systems, focusing on recommending interesting research papers from a user-centric perspective. He identifies the different tasks a recommender system could perform to assist the user, such as finding a starting point for research on a particular topic, and maintaining awareness of a research field. Recommendations are generated on the basis of citations in scientific papers [10].

Basu et al. (2001) focus on the related problem of recommending conference paper submissions to reviewing committee members [1]. They use a content-based approach to paper recommendation, using the Vector Space model with tf·idf weighting. Another related area of research is the development of recommender systems that employ folksonomies. Most of the work so far has focused on recommending tags for bookmarks. Jäschke et al. (2007), for instance, compared two different CF algorithms with a graph-based algorithm for recommending tags in Bibsonomy. They found that the graph-based algorithm outperforms the CF algorithms only for the top 3 ranks [6]. Mishne (2006) performs similar experiments when predicting tags associated with blog posts [11]. In our experiments we focus on CiteULike as the social reference manager. Capocci et al. analyze the small-world properties of the CiteULike folksonomy [2].

# 3. CITEULIKE

CiteULike is a website that offers a "a free service to help you to store, organise, and share the scholarly papers you are reading"[2] It allows its users to add their academic reference library to their online profile on the CiteULike website. At the time of writing, CiteULike contains around 885,310 unique items, annotated by 27,489 users with 174,322 unique tags. Articles can be stored with their metadata (in various formats), abstracts, and links to the papers at the publishers' websites. Users can also add reading priorities, personal comments, and tags to their papers. CiteULike also offers the possibility of users setting up and joining groups that connect users sharing academic or topical interests. These group pages report on recent activity, and offer the possibility of maintaining discussion fora or blogs. The full text of articles is not accessible from CiteULike, although links to online articles can be added.

## 3.1 Constructing a test collection

CiteULike offers daily dumps of their core database[2]. 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 of the other metadata described above, so we crawled this metadata ourselves from the CiteULike website using the article IDs. We collected the following five types of metadata:

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

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

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

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

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

As CiteULike offers the possibility of users setting up groups that connect users that share similar academic and topical interests, for each group we collected the group name, a short textual description, and a list of its members.

## 3.2 Characteristics of the collection

After crawling and data clean-up, our collection contained a total of 1,012,898 different postings, where we define a posting as a user-item pair in the database, i.e. an item that was added to a CiteULike user profile. These postings comprised 803,521 unique articles posted by 25,375 unique users using 232,937 unique tags. Metadata was available for 543,433 of the 803,521 articles[3]. CiteULike contained 1,243 different groups with 2,301 different users being a member of one or more groups, corresponding to 9.1% of all users. We did not crawl the full text of publications, but 33.7% of the articles included the abstract in their metadata.

# 4. RECOMMENDING USING CITEULIKE

McNee identifies eight different tasks that a recommender system could fulfill in a digital library environment [10]. Not all of these tasks are equally applicable in the CiteULike environment, and not all of them can be fulfilled using the collection we created. However, a social reference manager could arguably fulfill additional, new tasks not applicable in a digital library environment. In this paper we focus on the task of generating list of related papers based on a user's reference library. This task corresponds most closely to McNee's tasks of *Fill Out Reference Lists* and *Maintain Awareness* [10]. In contrast to McNee's approach of using citations, we use the direct user-item preference relations to generate our recommendations from.

## 4.1 Experimental setup

In order to evaluate different recommender algorithms on the CiteULike data and to compare the usefulness of the different information we have available, we need a proper framework for experimentation and evaluation. Recommender systems evaluation—and the differences with IR evaluation—have been addressed by, among others, Herlocker et al. [4, 5], the latter identifying six discernible recommendation tasks. The recommendation task we evaluate here is the "Find Good Items" task[4], where users are provided with a ranked list of recommended items, based on their personal profile.

Following common practice in recommender system evaluation [4, 5, 10], to ensure that we would be able to generate reliable recommendations, we select a realistic subset of the CiteULike data set by only keeping the users who have added 20 items or more to the personal profile. In addition, we filter out all articles that occur only once, since these items do not contain sufficiently reliable ties to the rest of the data set, and thus would only introduce noise

[2]See http://www.citeulike.org/faq/data.adp.

[3]The overwhelming majority of the articles with missing metadata were spam articles. How we detected this is beyond the focus of this paper.

[4]Also known as Top-$N$ recommendation.

for our CF algorithms. This procedure generates a set of 258,701 user-item pairs, with 2,539 unique users, and 87,908 unique items.

When generating predictions, we withhold 10 randomly selected items from each user, and generate predictions by using the remaining data as training material. As retrieval algorithms, recommender algorithms tend to be controlled by several parameters. One way of optimizing these parameters is by maximizing performance on a given data set. Such tuning, however, tends to overestimate the expected performance of the system, so in order to prevent this kind of overfitting we used 10-fold cross-validation [9].

We first divided our data set into a training and a test set by randomly selecting 10% of the users to be in our test set. Final performance was evaluated on this 10% by withholding 10 items from each user, and using the remaining profile items together with the training set to generate the recommendations for those 10%. In order to properly optimize parameters we divided our training set (containing 90% of the users) by randomly dividing the users over 10 folds, each containing 10% of the training users. Each fold is used as a validation set once, with 10 items being withheld for each validation fold user. The remaining profile items and the data in the other 9 folds are then used to generate the predictions. The final values for our evaluation metrics on the withheld items were then averaged over the 10 folds.

## 4.2 Evaluation

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 make up the relevant 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. While it is certainly possible and very likely that the recommendation lists contain other recommendations that the user would find relevant or interesting, we cannot know this without the user judging them. This means that because our relevance judgments correspond to items added to a user's profile, we can never have any items judged as being not relevant without user intervention.

Herlocker et al. [5] assesses the usefulness of different metrics for each of the six recommendation tasks they identified. For our "Find Good Items" task, they find that metrics taking into account the ranking of the items are most appropriate. We therefore evaluated our recommender system using Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and Precision @ 10 (P@10). In addition to these IR metrics, we also measured coverage: certain recommendation algorithms need enough data on a user or item for them to be able to reliably generate recommendations. Not all algorithms will be able to cover every user or item. We therefore measure user coverage (UCOV), which is the percentage of all users for which we were able to generate any predictions at all.

## 4.3 Algorithms

In the preliminary experiments described in this paper we compare three different CF algorithms. The term collaborative filtering was first used by Goldberg et al. [3] and describes a class of algorithms that, instead of looking at the content, use data about all users' preferences for items to generate recommendations for the so-called 'active' user. We use and briefly describe the two most simple variants: *user-based filtering* and *item-based filtering*[5]. In user-based filtering, the active user is matched against the database to find the neighboring users that the active user has a history of agreeing with. Once this neighborhood has been identified, all objects in the neighbors' profiles unknown to the active user are considered as possible recommendations and sorted by their frequency in that neighborhood. A weighted aggregate of these frequencies is used to generate the recommendations [4]. Item-based filtering turns this around by matching items against the database to find the neighborhood of similar items [13].

We test three different algorithms implemented in the freely available Suggest recommendation engine [7]. The first model is an item-based filtering approach that uses the cosine similarity metric to determine the similarity between items. Model 2 is an item-based algorithm that calculates similarity based on conditional probability. Finally, model 3 is a user-based algorithm that uses the cosine similarity metric to determine similarity between users. Item-based filtering tends to perform well when there are more users than items in the database whereas user-based filtering works better in the reverse situation. We therefore expect the user-based filtering algorithm to outperform the other two algorithms.

[5]See [14] for an overview of more sophisticated CF algorithms.

# 5. RESULTS

The CF algorithms we employ in our study have one important parameter that can be tuned: the neighborhood size $k$, i.e. the number of similar users used to generate the predictions. We tune this parameter for each of the models, using our 10-fold cross-validation setup described in Section 4.1, varying $k$ between 1 and 500, evaluating performance at each step. Figure 1 displays the effect of neighborhood size on the MAP scores of the three algorithms for $k$ up to 140; MAP values stay the same for $k > 140$. The other metrics show a similar trend over the different $k$ values.

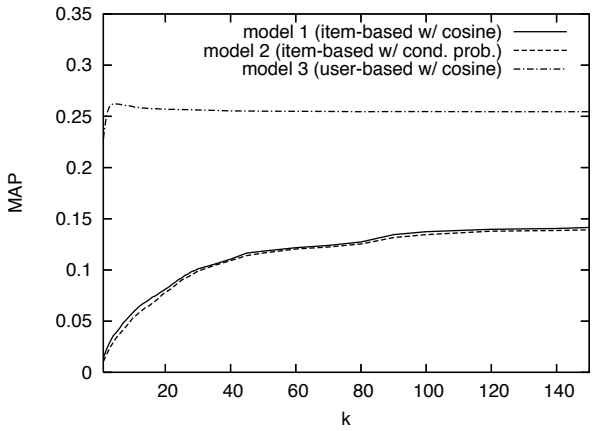


**Figure 1: The effect of neighborhood size $k$ on MAP for the three algorithms.**

The two item-based algorithms, models 1 and 2, both have an optimal $k$ of 500. Performance increases for Models 1 and 2 as $k$ increases, but the recommender implementation ran out of memory when we tried increasing $k$ beyond 500. The optimal value of $k$ for Model 3 lies between 4 and 8, with none of the differences in this range being statistically significantly different. We therefore picked a $k$ of 5 as the optimal value for Model 3. However, for all values of $k$ the user-based filtering algorithm significantly outperforms the item-based filtering algorithm ($p < 10^{-10}$). Final performance of the three models on the test set is displayed in Table 1. Model 3 outperforms the other models significantly on all measures ($p < 10^{-6}$) and shows acceptable performance for a preliminary approach.

# 6. TEMPORAL ANALYSIS

CiteULike's daily database dumps offer us the unique opportunity of analyzing the influence of growth of a social bookmarking

**Table 1: Results of our three CF algorithms on the data set.**

| | Model 1 | Model 2 | Model 3 |
|---|---|---|---|
| **MAP** | 0.1307 | 0.1344 | 0.2478 |
| **MRR** | 0.2622 | 0.3004 | 0.3278 |
| **P@10** | 0.1412 | 0.1451 | 0.2524 |
| **UCOV** | 92.09% | 92.09% | 99.60% |

service such as CiteULike on recommendation performance. The database dumps contain the time stamps for each posting; we used these to construct growth curves of the different algorithms from a temporal perspective. We gradually increased the size of our data set of user-item pairs by a month at a time and used the same setup and optimal parameters at each temporal step. The first item was added to CiteULike on November 4, 2004, giving us 37 months of data. We repeated these steps 10 times and calculated the average MAP scores.

Figure 2 shows the MAP scores for the three different algorithms plotted against the months in the CiteULike data set. Throughout the 37 months, the user-based filtering method always outperforms the other two item-based models. All CF algorithms achieve relatively high MAP scores in the initial months, after which performance remains low and erratic for the first two years. The last months of 2006 mark a turning point where a critical mass seems to have been reached. From that point on, recommendation performance starts to climb and even triples in the span of one year for all three algorithms. With the exception of the outliers in the first few months, this seems to be a clear illustration of the cold-start problem CF algorithms suffer from at a system-wide level: it is hard to generate recommendations for new items in the start-up phase, when there is not enough usage data about new items to make reliable correlations with other items [5].

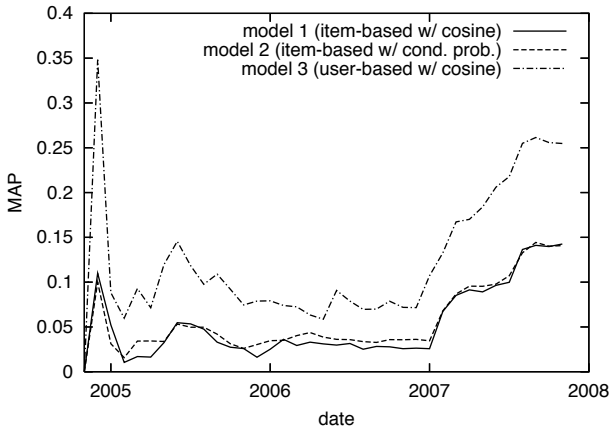


**Figure 2: Performance on the CiteULike data set over time.**

# 7. DISCUSSION & FUTURE WORK

In this paper we demonstrated how a social reference manager can be used as a test collection for recommending research papers. The data we collected represent many different aspects of both users and their annotations, and offers many opportunities for testing and combining different representations and recommender algorithms. We found that a user-based filtering algorithm yields the best performance. The likely explanation for this is that the data set contains a magnitude more items than users, which makes it both harder and slower to perform item-based filtering. Another interesting observation is that the optimal neighborhood size for item-based filtering is five users. This corresponds rather well to the average group size of 4.8 on CiteULike.

We also performed a temporal analysis on the data, and identified a clear cold-start problem for CF algorithms on the CiteULike. The results show that it took about two years for CF performance to start improving to a useful level. An interesting question is whether this two year period is specific to CiteULike or that a similar pattern would emerge when repeating these experiments on other social bookmarking websites. This is an important and interesting point for future work.

In other future work, we plan to experiment with and combine several different recommender algorithms, based on the different contextual and metadata information present in the CiteULike data set. Content-based algorithms should be able to take advantage of the full text of the documents and the metadata, while activity logs can be used to take recency effects into account. The CiteULike folksonomy offers another promising avenue of contextual recommendation material. The different algorithms based on the different contextual data will then be combined to determine the optimal recommendation process. Our end goal is to test our methods on users for one or two of the tasks identified by McNee et al. [10].

# 8. ACKNOWLEDGMENTS

The work of Toine Bogers is funded by the IOP-MMI program of SenterNovem / The Dutch Ministry of Economic Affairs, as part of the À Propos project. Antal van den Bosch is funded by NWO, the Netherlands Organization for Scientific Research.

# 9. REFERENCES

[1] C. Basu, H. Hirsh, W. W. Cohen, and C. Nevill-Manning. Technical Paper Recommendation: A Study in Combining Multiple Information Sources. *Journal of Artificial Intelligence Research*, 1:231–252, 2001.

[2] A. Capocci and G. Caldarelli. Folksonomies and Clustering in the Collaborative System CiteULike. eprint arXiv: 0710.2835, 2007.

[3] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

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

[5] 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.

[6] R. Jäschke, L. B. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag Recommendations in Folksonomies. In *Proceedings of PKDD 2007*, volume 4702 of *Lecture Notes in Computer Science*, pages 506–514. Springer Verlag, 2007.

[7] G. Karypis. SUGGEST Top-N Recommendation Engine. Available for download from http://glaros.dtc.umn.edu/gkhome/suggest/, 2000.

[8] P. Maes. Agents that Reduce Work and Information Overload. *Software Agents*, pages 145–164, 1997.

[9] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[10] S. McNee. *Meeting User Information Needs in Recommender Systems*. PhD thesis, University of Minnesota, June 2006.

[11] G. Mishne. AutoTag: A Collaborative Approach to Automated Tag Assignment for Weblog Posts. In *Proceedings of WWW '06*, 2006.

[12] M. Montaner, B. López, and J. L. de la Rosa. A Taxonomy of Recommender Agents on the Internet. *Artificial Intelligence Review*, 19(4): 285–330, 2003.

[13] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of Recommendation Algorithms for E-Commerce. In *Proceedings of EC '00*, pages 158–167, New York, NY, 2000. ACM.

[14] L. Si and R. Jin. Flexible Mixture Model for Collaborative Filtering. In *Proceedings of ICML '03*, pages 704–711. AAAI Press, 2003.