Digging Friendship: Paper Recommendation in Social Network

R. Dong and L.Tokarchuk
Queen Mary University of London
School of Electronic Eng. and Computer Science
Mile End Road, London E1 4 NS England
{ee08m010, l.n.tokarchuk}@elec.qmul.ac.uk

A Ma
MPI-QMUL Information Systems Research Centre
Macao Polytechnic Institute
Rua de Luis Gonzaga Gomez, Macao
{athen.ma@elec.qmul.ac.uk }

Abstract

Most paper recommender systems largely rely on explicit keywords provided by their users. This means that people are still find it challenging to find desirable papers from the hundreds of recommendations provided. Added personalization of recommendation would help to alleviate this problem. .Social networking has investigated the use of recommendation on a different basis than traditional recommenders. Friendships connect people together in a social network and thus contain various implied information about an individual. Social networks use this information to enable users to expand their social network, and thus tend to recommend new users or groups based on the existence of person or group amongst your friends, or friends of friends - s*ocial recommendation*. This type of information, while perhaps not appropriate for paper recommendation on its own, can help tailor recommendations toward the individual, rather than the user.

In this paper, we explore how we can refine paper recommendation using both traditional and social recommendation techniques. This paper first compares and evaluates several existing paper recommendation techniques, and then integrates the most effective one with social information. This information is gathered by the reading histories and recommendations from the active user's friends. Preliminary tests of this technique suggest that paper recommendations made by also taking friendships into account are not only accurate, but we feel more intelligent and better fitting user's preference.

1. Introduction

The Internet has given people opportunities to access research papers all over the world in a just a few minutes, which was impossible only a decade ago. However, mass availability leads to an overwhelming amount of information, leaving people to sift through millions of available choices. With rapidly increasing publications every year, it is impossible to cope with the published work in just a single narrow field, not to mention interesting interdisciplinary works.

Fortunately, recommender systems were built to deal with information overload. They have been applied to many domains with different techiques, for instance, GroupLens[1] and Ringo[2] apply Collaborative Filtering (CF) in Usenet new and music, respectively, Krakatoa Chronicle[3] system uses Content-based Filtering (CBF) on newspapers and Fab[4] applies the combined technique of CF and CBF to recommend web pages. Moreover, recommender system has been investigated in the domain of research paper as well, such as TechLens Project[5] uses CF to recommend research papers to their users.

As mentioned above, two kinds of techniques are normally used in recommender systems, namely, Collaborative Filtering (CF) and Content-based Filtering (CBF). The term CF was first introduced by Goldberg in [5], it recommends items to the active user if similar users liked those items. The intuition behind this is that if users have similar taste in the past, they tend to agree in the future as well [11].

CBF techniques have been well studied in the information retrieval (IR) field. They recommend items based on the keywords contained in the user profiles or

[1] Designed by Resnick, P.; Iacovou, N.; Sushak, M. and Bergstrom, P. 1994, http://www.grouplens.org/
[2] Designed by P.Maes and U. Shardanand, 1995, http://ringomo.com/
[3] Designed by T. Kamba; K. Bharat and M.Albers, 1995
[4] Designed by M. Balabanovic and Y. Shoham, 1997
[5] Designed by S. McNee, I. Albert, et al. 2002, http://techlens.cs.umn.edu/tl3/

inputted by the user. There are many ways to measure the similarity between contents [2] [9]. The CBF technique applied in this research is TF-IDF, which has been widely used in many domains and is considered as a successful technique for text similarity [7]. The term TF-IDF stands for term-frequency inverse-document-frequency, which combines the frequency of terms in documents with the distribution of the terms in the whole collection of documents. In other words, the importance increase proportionally to the number of times a word appears in the documents but is offset by the frequency of the word in corpus. Theoretically, documents with high number of similar words or discriminating words should be the most similar to the query.

CF has several advantages compared to the CBF [3][6], such as:

1. *Independence of content.* Because the items are rated by humans, there is no constrains for what kinds of items could be evaluated. Movies, newspapers, or people all can be used as the domain of CF.
2. *Enhancing the quality of the recommendations.* This is due to the involving of human evaluation of items. Humans can assess whether the paper is authoritative or well-written, which is a hard work for computers.
3. *The ability to give serendipitous recommendations.* Because the similarity is measured between people rather than the items, users may get unexpected recommendations from similar users.

On the other hand, CF has its own drawbacks [3] [6]:

1. *First-rater problem*, which is that an item can not be recommended until at least a user rates it.
2. *Start-up problem*, that is, CF can not give satisfactory recommendations for user with few ratings since he could not be placed in a good neighborhood to be similar to any others.
3. *Sparsity problem*. Because in the real word, users are very likely to rate only small part of the existing items, which leads to the lack of overlap of tastes and therefore make it difficult to create neighborhoods.

CBF also has its own advantages, such as:

1. *Does not suffer from first-rater problem*, since it recommends items if their text share any common words with user's profile or keywords.
2. *Does not suffer from sparsity problem*, because for most items, the text similarity to the user profile can be computed.

However, it has several shortcomings [3]:

1. One is that it does not take the quality such as authoritativeness, style into consideration because most text analysis techniques are based solely on word analysis.
2. Another one is the over-specialization problem which refers to the fact that it recommends items by analyzing the content of the texts without spreading between other subjects. For instance, it may not consider two different texts that use the word "car" and "automobile" respectively as the similar items.

In addition, with the increasing popularity of social network, recommender systems have begun to be introduced in social networks. More and more people are willing to share their personal lives on social network sites, thus making it possible to make more personalized recommendation by using the information contained by those relationships. In our research, we use friendship to refine the recommendations to fit every individual user since we believe that friends tend to have similar tastes, and therefore recommend papers that have already been read by friends of the active user over those that have not.

The rest of this paper is organized as follows: Section 2 will discuss several existing paper recommendation techniques implemented in our research. Section 3 will evaluate and compare those techniques to find the most effective one.

2. Algorithms

To find the most efficient recommendation algorithm, we present seven different algorithms, which each receive one paper as input and generate a list of papers recommendations as output.

Classic implementations of CF and CBF are included as a baseline comparison. And based on the CBF algorithm, both CBF Separated and CBF Combined algorithms are developed to spread search space. In addition, three hybrid algorithms, namely CF-CBF Separated, CBF Combined-CF and Fusion algorithms, are also implemented to let CF and CBF complement each other and therefore can overcome some of their shortcomings, such as first-rater and over-specialization problem. The algorithms are then evaluated using the Citeseer[4] database, an online database of computer science research papers.

While CBF was applied over the text of the papers, CF recommended items based on the similarity among users. Normally, this similarity is calculated based on user profiles, which is represented by a rating matrix. Therefore, in order to perform CF in the domain of research paper, a rating matrix needs to be created first. To do that, we chose to follow the same approach used in [9] which considers the papers as "users" and their citations as "items" they have

rated. This approach does not suffer from startup problem since the rating matrix is populated from citations of papers. It also frees the system from rating consistency because the ratings are not received from actual human beings, who may have different rating criteria from each others.

1. Collaborative Filtering Algorithm:

The most popular algorithms used in CF are the neighborhood-based algorithms. In which the neighbors of the active user are calculated based on their similarity to the active user, and then the predictions are generated by calculating the weighted aggregate of their ratings. There are many techniques to perform neighborhood-based CF: user-user [6], item-item [8], and co-occurrences [9].

In our research, the CF algorithm we use is the standard k-nearest-neighbor user-user CF algorithm. As mentioned above, a paper is considered as a "user" and its citations as "items" rated. Therefore, this algorithm takes the citations of the active paper as input and generates a list of recommendations as output, respectively.

2. Content-Based Filtering Algorithm:

CBF algorithms recommend papers based on their content similarities. For a given paper, the most similar papers are recommended. The method applied to calculate text similarity is TF-IDF, as previously described in section 1.

Pure CBF is straightforward as its name suggested, it searches for similar papers according to their text similarity to the active paper and then the most similar ones are recommended.

3. CBF-Separated Algorithm:

This algorithm is built upon the pure CBF algorithm [1]. It explores not only the text of the paper, but also the text of all the papers it cites. It searches for similar documents to the paper itself and for all citations made in the original paper.

For example, as shown in figure 1, for the current paper P, this algorithm generates a list of similar papers (R_{1p} ..... R_{mp}), where m is the number of recommendations. And for every citation C_1 to C_n, where n is the number of citations in the current paper p, the algorithm also generate a list of similar papers ((R_{1C1} ... R_{mC1}) ... (R_{1Cn} ...R_{nCm})). Then all lists are merged into one single list sorted on the similarity coefficient returned. Finally, papers with highest similarity scores are recommended to the user.

One advantage of this approach is that it tends to reduce over-specialization since it expands the search space beyond similarity to the current paper to include similarity to citations as well.

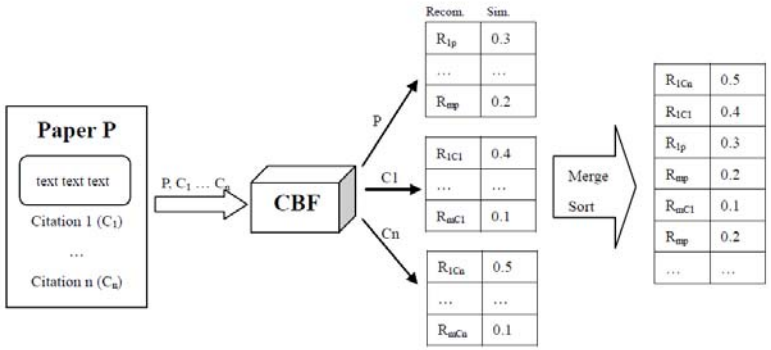


Figure 1. CBF-Separated Algorithm

4. CBF-Combined Algorithm:

CBF-Combined algorithm [1] is another extension of pure-CBF. Instead of generating lists of similar papers for every citations as in CBF-Separated, it merges the text of the active paper and the text of all the citations together into one large chunk of text. After that, this large text is submitted as the query to pure-CBF and the most similar papers to the combined text are returned as the recommendation. This process is shown in the following figure 2.

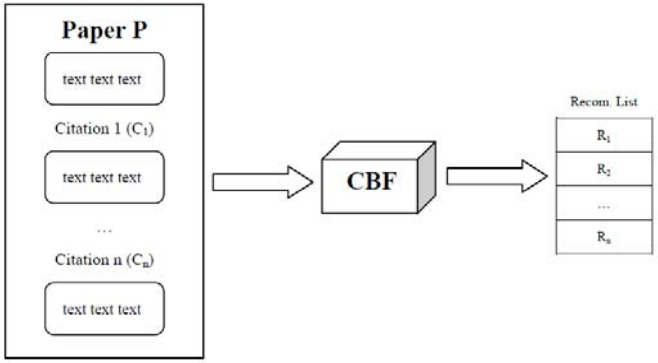


Figure 2. CBF-Combined Algorithm

5. CF-CBF Separated Algorithm:

Both CF and CBF algorithms have their own advantages and disadvantages. As mentioned in Section 1, CBF can suffer from over-specialization problem, while CF the first-rater and sparsity problem. Combining both algorithms together can overcome their shortcomings, at least partially.

As shown in figure 3, in CF-CBF Separated, the CF (algorithm 1) is run first in order to generate a list of recommendations,. CBF-Separated (algorithm 3) is then applied to give further recommendations based on the list generated from CF. In other words, the CBF module recommends a set of similar papers for every recommendation given by CF to the active paper. Because the recommendations given by CF module are in order, the recommendations generated by the CBF module have to be scaled by this ordering. To achieve that, the CBF recommendations are weighted accordingly, with the first set generated from the top CF recommendation receiving weight 1 and the following sets' weights decreased by 0.05 accordingly. Then the similarity scores of CBF recommendations are multiplied by these weights in descending order. The following formula shows the above process:

$$S(r_k) = S(r_k) \times (Wp_{(j-1)} - \Delta d) \qquad (1)$$

where r_k is the k th recommendation in the CBF list, p_j is the j th recommendation in the CF list, $Wp_{(j-1)}$ is the weight given to the previous recommendation of CF, Δd is the decremented weight factor and $S(r_k)$ is the similarity of r_k with p_j.

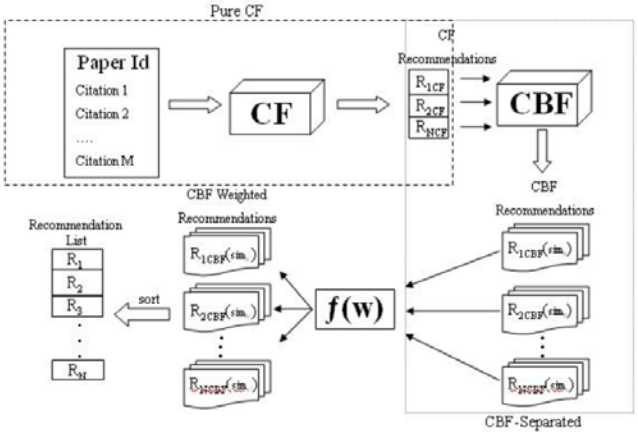


Figure 3. CF-CBF Separated Algorithm

6. CBF Combined-CF Algorithm:

This algorithm still attempt to address the shortcomings of CBF and CF by combining them together. Different from CF-CBF Separated, as shown in figure 4, this algorithm applies CBF-combined first and then CF. In other words, the recommendations given by CBF-Combined are used to augment the citations of the active paper, and then the CF algorithm uses the active paper along with its augmented set of citations to generate further recommendations.

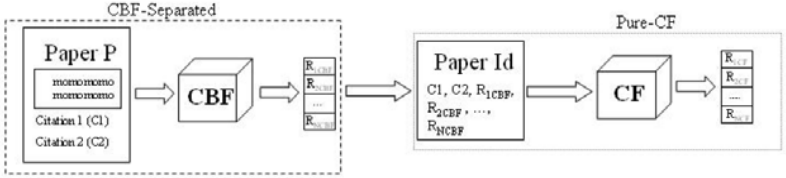


Figure 4. CBF Combined-CF Algorithm

7. CBF- CF Parallel Algorithm:

This algorithm runs both CF and CBF modules in parallel and generates the recommendation list by merging the results from both modules as described in [1]. In addition, to sort the recommendations in the right order, the recommendations by both algorithms are passed through an ordering function before being merged to the final list.

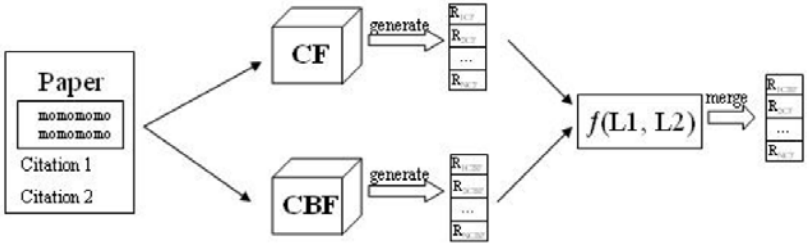


Figure 5. CBF- CF Parallel Algorithm

The ordering function works as follow: every recommendation that shows in both modules' recommendation lists is added to the final list with a rank score, which is the summation of the ranks of recommendation in their original lists. Then the final list is sorted in ascending order based on these scores. For example, a paper that is ranked 3rd from the CF module and 2nd from CBF will

receive a score of 5. Item with lower score goes closer to the top of the final recommendation list. Other recommendations that don't appear in both CF and CBF lists are alternatively added to the final list. The general process of CBF-CF Parallel Algorithm is shown in figure 5.

3. Evaluation

In order to evaluate the above algorithms, a dataset created from papers extracted from CiteSeer, an online database of computer science research papers, was used. The dataset was limited in two ways: first, papers that cite fewer than 2 other papers are removed from the dataset since loosely connected papers may introduce noise to the dataset [1]. Second, the citations not included in the dataset as papers are removed. By doing this, every citation in the dataset is also a paper in the dataset and therefore both CF and CBF would be able to analyze every paper in the dataset[6]. After this process, there are 68,625 papers in our pruned dataset.

To follow the methodology "Leave one out" [9], the dataset is divided into training and testing datasets at a 90% to 10% ratio firstly. Then one citation is randomly removed for every paper in the testing dataset. This process is repeated ten times in order to perform 10-fold cross validation.

However, the recommender may give papers that didn't exist at the publication time of the active paper as recommendations. Since we measure the performance of algorithms by looking at the rank of the removed citations in the recommendation list and the citations of a paper must already exist at the time the paper is written in order to be referred. Therefore, to handle this problem, we filtered out recommendations that are published later than the active paper. It is important to point out that the recommender could recommend papers that are very similar to or even better than the removed citation and this may diminish the performance of algorithms because these papers will appear before the removed citations in the recommendation list. Even though there is a possibility, we still expect the removed citation to be recommended.

The "hit-percentage (HP)" is used as a metric to measure the percentage of time the recommender algorithm correctly recommends the removed citation. The rank where the removed citation was found in the recommendation list is also measured. Because the rank is important to users, the analysis is segmented into bins based on rank where lower is better (e.g. recommendation in the top10 bin is better than that in top40 bin). Recommendations beyond the 40th are considered "all" (41-100) for the reason that users are not likely to see items recommended beyond this position. Table 1 shows the rank bins.

Rank of the citation found	Bins
1	Top1
1-10	Top10
1-20	Top20
1-30	Top30
1-40	Top40
1-N	All

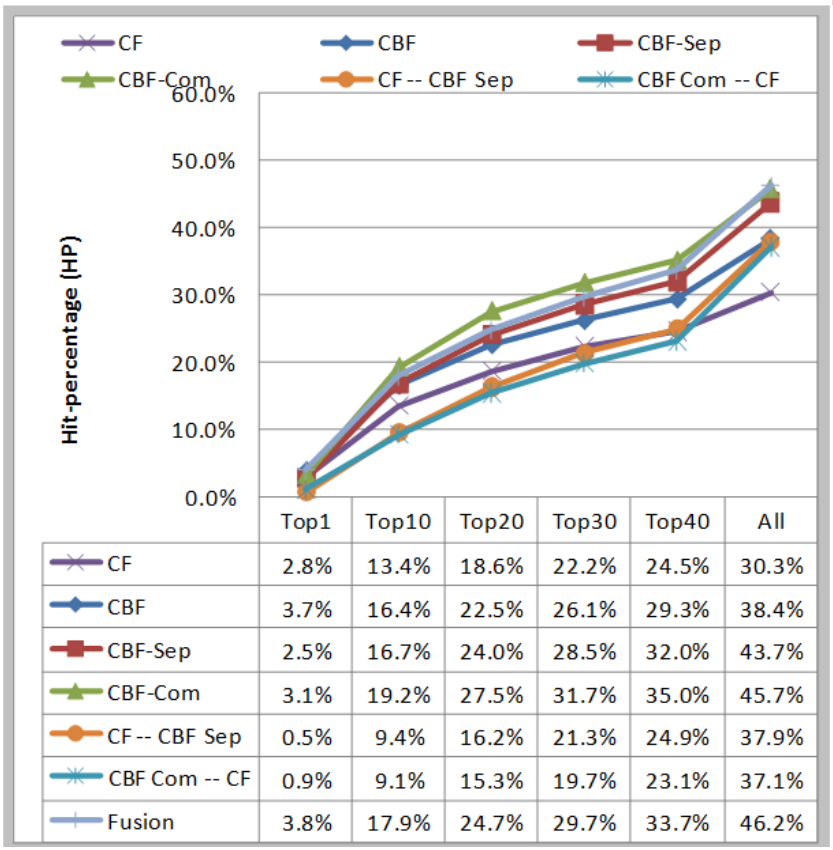
Table 1. Rank Analysis

We are following the two criteria described in [1], that is, the first one is focus on the performance in the *Top10* as we think users like the best recommendations first. The second one is the ability to give the recommendation of removed citation in despite of its rank, which is measured by *All*. If the *Top10* of one algorithm is better than another algorithm but the *All* is not, or vice-versa, the results of *Top1* decide the best algorithm. The best algorithm will be selected to integrate with social information collected from social network sites.

As seen in figure 6, pure CF performs the worst in *All*, this is probably due to sparsity problem of CF algorithm itself as there's lack of overlap among citations of papers to create good neighborhood.

For the three CBF algorithms, pure CBF works well in *Top1,* while CBF-Separated worst among three of them. The CBF-Combined performs best from *Top10* to *Top40* in our dataset, while the performance of CBF-Separated is quite good as well. This suggest that spreading search space by including the text of citations as search query or similar papers to the citations as recommendation can reduce the over-specialization problem of CBF itself.

[6] Since algorithm 3, 4, 5 and 6 need to access the text of the citations as well, this requires the citation to be included in the dataset.



	Top1	Top10	Top20	Top30	Top40	All
CF	2.8%	13.4%	18.6%	22.2%	24.5%	30.3%
CBF	3.7%	16.4%	22.5%	26.1%	29.3%	38.4%
CBF-Sep	2.5%	16.7%	24.0%	28.5%	32.0%	43.7%
CBF-Com	3.1%	19.2%	27.5%	31.7%	35.0%	45.7%
CF -- CBF Sep	0.5%	9.4%	16.2%	21.3%	24.9%	37.9%
CBF Com -- CF	0.9%	9.1%	15.3%	19.7%	23.1%	37.1%
Fusion	3.8%	17.9%	24.7%	29.7%	33.7%	46.2%

Figure 6. Results of all algorithms

Surprisingly, combining algorithms by applying them alternatively dose not give a better result, especially in *Top1*, only 0.5% for CF-CBF Sep and 0.9% for CBF Com-CF. This does not necessarily mean that these two algorithms are not better than other ones, this result probably due to the inter-connections among papers in our dataset are too loose to create good neighborhood for CF, which may undermine the total performance of two algorithms. Or the recommendations generated are more similar to the active papers than the removed citations, which decrease the rank of removed citations. However, it is too difficult to test in offline evaluations without human interaction.

Finally, the fusion algorithm performs best in *Top1* and *All*.

According to the criteria described above, CBF-Combined performs best in *Top10* while fusion is the best in *All*. Therefore, *Top1* decides the best one. That is, the fusion algorithm is chosen to be integrated with social information.

4. Integrating with Social Information

As we believe that friends have similar taste and tend to share the similar interest in further as well, papers read by active user's friends may be more attractive than those generated by computer. To refine the paper recommender system designed above, we integrate the recommender with reading histories and paper collections of active user's friends. By re-ordering the recommendations, we expect the recommender to be more personalized and intelligent for the involving of human interaction.

The approach designed to be test online in social network site is as following:

First, the search histories and paper collections of active user's friends are gathered to generate a paper list L. The papers are sorted in the list according to their popularity among friends, that is, the more friends had read that paper, the nearest to the top of the list. Second, the recommendations to the selected paper (as described above, a paper is chose as the input of algorithms) are generated by Fusion algorithm. Then, list L is compared with those recommendations to check whether they contain same papers, that is, those papers that have already read by friends as well as recommended by the Fusion algorithm. If so, the recommendations are re-ordered to prioritize those papers in the top of recommendations. After that, both the original and re-ordered recommendations are presented to the active user. Finally, the choices of user are gathered and recorded to further evaluate this new approach. In addition, if none of the paper in the recommendation list has read by any friends of active user, that is, the original and re-ordered recommendation list are exactly the same, the choices of the user will not be recorded to eliminate the possibility of random choice made by user.

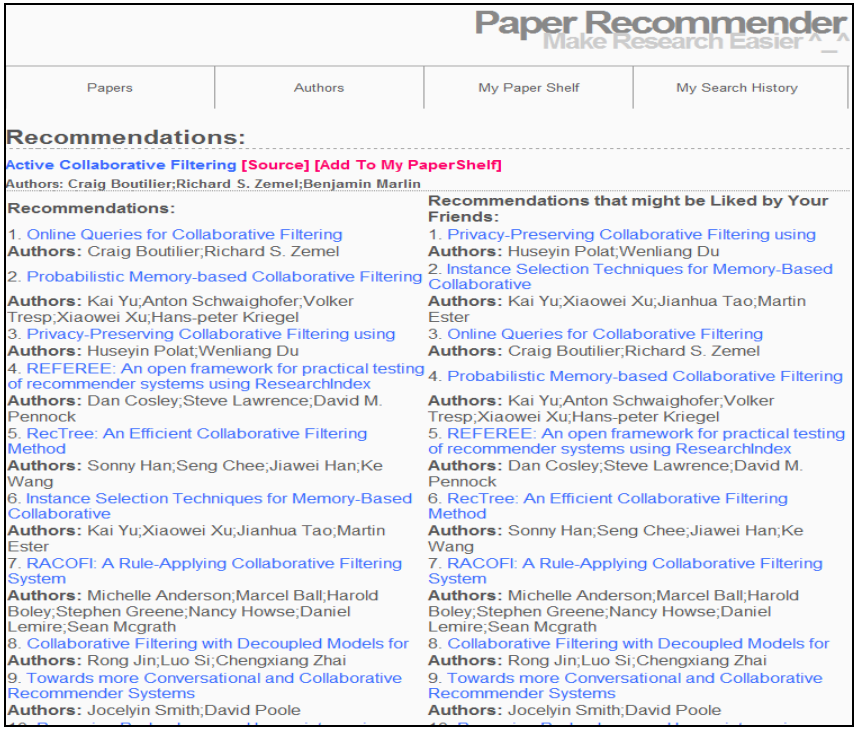


Figure 7. Paper Recommender

As seen in above figure 7, the column on the left is the original recommendation list while the right one takes friends' searching histories and collections into consideration. The first two papers in the re-ordered list (right list) are shown in the top since they are shared by the friends of active user. The paper "Privacy-Preserving Collaborative Filtering using" appears before the paper "Instance Selection Techniques for Memory-Based Collaborative" because it have been read by more friends or at least the same number of friends as the second one.

5. Further Issue

As our goal is to facilitate researchers in their studies, our next step is to launch the paper recommender system in the popular social network site, such as Facebook, to facilitate more researchers as well as other users and collect more user feedbacks for the further evaluation since the preliminary test suggests that users prefer recommendations made by taking friendship into consideration to the traditional ones. Another possible extension is to investigate other relationships in social network that might help make the recommendation more personalized, such as the groups in social network sites, as the members of the group share some properties and may have similar interests or backgrounds, which could be used to improve the recommendations as well.

References

[1] Porto Alegre, "Combing Collaborative and Content-based Filtering to Recommend Research Papers", PhD Thesis, 2004.

[2] R. Baeza-Yates and B. Ribeiro-Neto, "Modern Information Retrieval", Addison-Wesley, Wokingham, UK, 1999.

[3] Marko Balabanović and Yoav Shoham, "Fab: Content-Based, Collaborative Recommendation", *Communications of the ACM*, New York, 1997

[4] Kurt D. Bollacker, Steve Lawrence and C. Lee Giles, "A System For Automatic Personalized Tracking of Scientific Literature on the Web", *in International Conference On Digital Libraries*, Berkeley, CA, 1999.

[5] D. Goldberg, D. Nichols, B. Oki and D. Terry, "Using collaborative filtering to weave an information tapestry", in *Communications of the ACM*, New York (USA), 1992.

[6] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers and John Riedl, "An Algorithmic Framework for Performing Collaborative Filtering", in *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*. Berkeley, CA, 1999.

[7] H. Kroon and E. J. H. Kerckhoffs, "Improving Learning Accuracy in Information Filtering", *in International Conference on Machine Learning*, Bari, Italy, 1996.

[8] Bradley N. Miller, Joseph A. Konstan, and John Riedl, "PocketLens: Toward a Personal Recommender System", *ACM Transactions on Information Systems (TOIS)*, Volume 22, Issue 3, Pages: 437 – 476, July 2004.

[9] Sean McNee, Istvan Albert, Dan Cosley, Prateep Gopalkrishnan, Shyong K. Lam, Al Mamunur Rashid, Joe Konstan, and John Riedl, "On the Recommending of Citations for Research Papers", in *Proceedings of ACM 2002 Conference on Computer-Supported Cooperative Work*, New Orleans, LA (USA), 2002.

[10] Gerard Salton and Chris Buckley, "Term Weighting Approaches in Automatic Text Retrieval", *Information Processing and Management*, 1988.

[11] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: an open architecture for collaborative filtering of netnews", in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, North Carolina (USA), 1994.