On the Recommending of Citations for Research Papers

Sean M. McNee, Istvan Albert, Dan Cosley, Prateep Gopalkrishnan,
Shyong K. Lam, Al Mamunur Rashid, Joseph A. Konstan, John Riedl

GroupLens Research Project
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455 USA
{mcnee, ialbert, cosley, prateep, lam, arashid, konstan, riedl}@cs.umn.edu

ABSTRACT

Collaborative filtering has proven to be valuable for recommending items in many different domains. In this paper, we explore the use of collaborative filtering to recommend research papers, using the citation web between papers to create the ratings matrix. Specifically, we tested the ability of collaborative filtering to recommend citations that would be suitable additional references for a target research paper. We investigated six algorithms for selecting citations, evaluating them through offline experiments against a database of over 186,000 research papers contained in ResearchIndex. We also performed an online experiment with over 120 users to gauge user opinion of the effectiveness of the algorithms and of the utility of such recommendations for common research tasks. We found large differences in the accuracy of the algorithms in the offline experiment, especially when balanced for coverage. In the online experiment, users felt they received quality recommendations, and were enthusiastic about the idea of receiving recommendations in this domain.

Keywords

Collaborative Filtering, Recommender Systems, Citation Graphs, Social Networks, Digital Libraries, ResearchIndex

INTRODUCTION

People face the problem of information overload every day, a problem that is only getting worse. As more and more people publish information on the World Wide Web, it becomes increasingly difficult to find needed information quickly.

By recommending items to users based on previously expressed user preferences, recommender systems help users navigate and control complex information spaces. The MovieLens recommender (www.movielens.org), for example, helps users of the system find movies to watch solely based on their opinions of films.

Collaborative Filtering (CF) is a widely used technique in recommender systems. CF works by matching users in a system based on the similarity of each user's past preferences. Each user has a 'neighborhood' of other users with similar opinions about items in the system. This neighborhood can be used to generate recommendations by suggesting items to the user that he has not viewed but that his neighbors have viewed and rated highly.

Collaborative filtering has several limitations. One of the most important is the *startup* problem [15]. When a CF system is first created, there are many items in the system, few users in the system, and no ratings. Without ratings, the system cannot generate recommendations and users see no benefit. Without users, there is no way for new ratings to be entered into the system. When applying CF to a domain, it is valuable to seek preexisting data that can be used to seed such a database of ratings. In the case of MovieLens, the freely available EachMovie dataset was used to "jump start" the system [7].

Collaborative filtering makes use of the relationships found between people, an approach that fits nicely with the idea that humans are fundamentally social creatures. One consequence of our sociability is that in our interactions with other people, we create social networks. These networks can be personal, professional, or political, can have varying importance on people's lives, and have been studied in detail by sociologists, psychologists, and physicists, among others [17]. While studying direct human-human social networks is extremely important, it is also worthwhile to study the social networks of artifacts, especially when these networks directly relate people and their artifacts together.

We are interested in how CF can be applied in one of these social networks of artifacts: the network of research papers. The citations between research papers form a graph that can be viewed as a social network known as a *citation web*. For any given paper, it is possible to follow the citation web to see what papers cite it and what papers are cited by it. This data can be mapped onto a CF framework and used to help overcome the startup problem [15].

Finding Research Papers

Searching for related work can be tedious and it is possible to miss important developments in areas outside a researcher's specialty. Many searches for related work consist of starting with a small number of initial papers and navigating the citation web near those papers. ResearchIndex [12] provides an interface to the citation

CSCW'02, November 16–20, 2002, New Orleans, Louisiana, USA.

web for computer science research papers. Search engines such as Google and AltaVista also help researchers find related work.

With these strategies, however, it is likely that some important related work will be missed: ResearchIndex is unable to effectively scan the entire citation web to find connections between papers[1]. Search engines, while able to scan all documents for relevant text, do not use semantic information, such as paper citations, in their results. We hypothesize that recommender systems, especially those based on collaborative filtering, will discover more powerful relationships among the citation web of research papers than the above methods for searching.

We chose to use ResearchIndex as a test bed for our exploration as it provides several advantages. ResearchIndex has spent years developing automatic citation indexing techniques that work on the domain of research papers [11]. They have demonstrated effective citation processing heuristics both for extracting citations from research papers, and for matching those citations with similar citations in other research papers. They have become a frequently visited and valuable resource to the research community.

Contributions

We provide three contributions to the fields of recommender systems and collaborative filtering.

First, we demonstrate four ways to apply collaborative filtering to the domain of research papers using the citation web that exists between papers. Several of these algorithms differ from conventional approaches to collaborative filtering. None of these approaches suffer from the collaborative filtering startup problem.

Second, we provide an offline experimental evaluation of the predictive quality of the four collaborative filtering-based algorithms along with two other recommendation algorithms in the domain of research papers.

Third, we provide an online experimental evaluation of user experiences with these six recommenders. We assess user perceptions of the relevance and novelty of the resulting recommendations, overall user satisfaction, and user opinion of the usefulness of the resulting system.

The outline of the rest of the paper is as follows. We first discuss related work in the fields of collaborative filtering, citation indexing, and social networks. We then talk about the citation web of research papers, and how we implemented our CF algorithms in that domain. Next, we describe our experiments, starting first with the specific algorithms we tested and continuing with detailed descriptions and analyses of both our offline and online experiments. We conclude with a discussion of the applicability of collaborative filtering to the domain of research papers.

RELATED WORK

Collaborative Filtering

In the field of collaborative filtering, both Herlocker et al. [7] and Breese et al. [1] have provided overviews and frameworks for evaluating CF algorithms. Many algorithms beyond the original *k*-nearest neighbor algorithm [15] have been proposed and used for collaborative filtering. These include item-based algorithms [16] and model-based algorithms such as Bayesian networks [1] and clustering [1]. Researchers have experimented with CF systems in a wide variety of domains, including Usenet news [15, 19], jokes [6], movies [7, 8] and music [18]. Collaborative filtering has succeeded in helping users in all of these domains.

ReferralWeb combined collaborative filtering, searching, social networks, and social networks of artifacts to create a recommender system to refer people with common interests to each other inside a pre-existing social network [10]. Our work extends ReferralWeb by exploring ways to directly apply CF to social networks themselves.

Most CF domains have independent items with relatively thin relationships to each other and little pre-existing ratings data. Research papers start with the rich web of citation relationships among papers. Applying CF to this domain successfully requires that the algorithms be modified to interpret the citation web data effectively.

Citation Indexing

By introducing automatic citation indexing [11], ResearchIndex was able to quickly create a large online citation web of Computer Science research papers [2, 12]. Automatic citation indexing works by using a series of heuristics to process documents. We are exploring techniques through which collaborative filtering may be able to improve the utility of citation indexing systems such as ResearchIndex.

Woodruff et al. have a recommender algorithm which "fuses text and citation data" to create recommendations inside a single digital book [20]. Hybrid filtering algorithms such as this, which combine both semantic and collaborative information, may have great potential applied to the field of research papers. However, we do not study such algorithms here.

Social Networks

Using the references found in research papers, it is possible to create citation webs that reflect professional social networks between researchers. Librarians and information professionals have studied the creation of these webs and ways to index them for years [3, 5]. In particular, many people have studied the connections between research papers and authors of research papers [13]. We are adding to this work by investigating how research papers directly relate to each other as opposed to the relationships that exist between papers and authors, and how these paper-to-

[1] Citations may be undercounted in ResearchIndex due to imperfections in the process of automatic citation indexing. Citations may also be undercounted due to copyright restrictions.

paper relationships can be exploited to create a system to recommend papers to authors.

RECOMMENDER SYSTEMS AND RESEARCH PAPERS

We draw a subtle but important distinction between the idea of a citation and that of a paper. A citation represents a research paper for which we only have a reference (i.e., a paper has cited this citation, but we do not possess the paper which corresponds to the citation). A paper is a citation for which we have access to the full text, including the paper's citation list. Thus, for a paper we have a listing of all the citations that it references, some of which *may* also be papers in our dataset but all of which *must* be citations in our dataset. It should be noted that a paper can exist without a citation. It would imply that the full text of the paper is in the system, but there is insufficient knowledge of its publication status, authors, etc. to create a citation.

Integrating CF into the domain of research papers

Standard CF algorithms work by viewing a dataset as a ratings matrix. Columns of the matrix represent 'items' in the CF environment, while rows represent users. Each entry in the matrix is the user's rating of a particular item. These ratings are gathered from users either implicitly, such as through purchase records and browsing history, or explicitly, by asking users to rate the items. Collaborative filtering predicts which values would appear in blank spots in this matrix by comparing the similarities of either the rows (users) or the columns (items) in the matrix.

In order to perform collaborative filtering on the domain of research papers, we need to map the citation web onto a collaborative filtering ratings matrix. There are several ways to create a collaborative filtering ratings matrix from the citation web between research papers.

The first way is to consider an analogy to MovieLens and other current collaborative filtering systems. In such a system, citations would be the 'items' in the matrix, while real people would be the 'users' who rate papers. This approach does not use the citation web and would suffer from the startup problem while collecting an initial set of ratings. We instead chose to directly mine the citation web.

An alternative approach is to make paper authors the 'users' and keep citations as 'items.' In this ratings matrix, each author would "vote" for the papers that she has cited. By using the citation web to populate the matrix, this mapping does not suffer from the startup problem. This is the method that has been explored before by Kautz et al. [10] and Newman [13].

This approach suffers from a generality problem. Many authors have written papers in several different fields over the course of their careers. For example, Ben Shneiderman has written many papers about human computer interaction and user interface design. However, he started his career with papers about FORTRAN and has also written several papers about Jewish heritage and history. With many prolific authors in the system such as Ben Shneiderman, it might be difficult to find a set of authors that would describe your information need. Imagine a user submitting as input a list of papers on user interfaces, and receiving recommendations for papers on Jewish history. This *serendipity* is useful in many domains, exposing users to items they might not otherwise consider, but may not be as useful in the context of research papers.

Instead of using an author's list of citations, we chose another mapping which uses the citation lists from individual papers. Here, a paper would represent a 'user' in the matrix and a citation would represent an 'item.' Each paper would then vote for the citations found in its references list. This mapping uses the citation web to populate the ratings matrix and therefore does not suffer from the startup problem. This mapping also does not suffer from the loss of specificity that could occur with the author-citation mapping.

By having the 'users' be papers, we are guaranteed that every 'user' in the system provides ratings information in the form of their votes. This allows us to create our ratings matrix without any startup problems. These 'users' will also never add more votes after the paper is first entered. This is in contrast to most CF environments where it is expected that users will add more ratings information over time. It is also expected that the citation lists will be of high quality, which frees the system from ratings consistency and 'rogue user' issues. This is the mapping that will use.

There are other mappings worth considering. For instance, there is the mapping where both the 'users' and 'items' in the ratings matrix are citations. In this matrix, a rating between the two would be some measure of co-citation between the two elements. A basic co-citation metric is a count of the number of times the two citations appear together in a references list on a paper. This measure of co-citation is related to the item-item collaborative filtering algorithm [16], which has been very successful in other domains. We include a co-citation algorithm as an alternative in our experiments.

Recommender Algorithms

Each of the six algorithms below receives a "basket" of citations as input and returns a ranked list of recommended citations. In our experiments, the basket consists of the citations made by a "target paper" for which we recommend other citations the author may wish to know about. The algorithms do not recommend items in the basket. The Google and Citation Graph Search algorithms use additional information (the title and abstract of the target paper) not used by the CF-based methods.

Collaborative Filtering Algorithms

We use the following four CF algorithms in our experiments. There are many other possible CF algorithms that we chose to not include (e.g. see [1, 7]).

Co-Citation Matching

Co-citation Matching works by counting co-citations. For each citation in the basket, the algorithm counts the number of times other citations were co-cited with it. The

algorithm recommends citations with the highest total co-citations summed over all of the basket items.

User-Item CF

User-Item CF is the original *k*-Nearest Neighbor CF algorithm [15]. Given the ratings matrix discussed earlier, the User-Item algorithm compares papers (rows) in the matrix to create a neighborhood of the most similar papers to the target paper. Since citations are binary (either a paper is cited or not), the algorithm uses a cosine similarity metric. The algorithm counts the number of times neighbors make a citation, with the count weighted by the similarity of each neighbor to the target paper. The algorithm recommends citations with the highest weighted counts. We used the freely available Suggest recommendation engine for our implementation of this algorithm [9].

Item-Item CF

Instead of building neighborhoods among users, Item-Item CF 'flips' User-Item collaborative filtering by comparing similar *items* [16]. The Item-Item algorithm compares citations (columns) in the ratings matrix to create a neighborhood, an 'item' neighborhood, of the closest citations to each citation in the basket. Again, we use Suggest for the implementation with a simple cosine similarity metric.

Naïve Bayesian Classifier

The Naïve Bayesian classifier [1, 4] calculates probabilities that any given citation in the dataset is related to the input basket. The algorithm sorts the citations by probability and recommends citations from highest to lowest probability. The classifier is trained on citation lists from the dataset. Even in domains where the naïve Bayes principle does not hold, naïve Bayesian classifiers still work remarkably well [4].

Non-Collaborative Filtering Algorithms

There are many other algorithms we could use as baselines. We chose two algorithms that we feel approximated how researchers look for relevant citations.

Localized Citation Graph Search

The Citation Graph Search uses keyword similarity between the target paper's abstract and the titles of papers nearby in the citation graph. For a target paper, it makes a list of papers that cite the target, papers co-cited with the target, and papers cited by items in the basket. The algorithm uses TF/IDF term weighting to match the titles of these nearby citations against the title and abstract of the paper in question, and returns recommendations sorted by relevance.

Keyword Search ('Google' Baseline)

The title of the paper in question is sent as a query to the Google search engine, restricted to results that appear in ResearchIndex. The algorithm recommends papers in the order returned by Google.

EXPERIMENTS

We performed two different experiments to see how the six algorithms would perform in the domain of research papers. In the offline experiment, we were interested in studying the ability of collaborative filtering methods to find missing citations. In the online experiment we used human subjects and focused on collecting user opinion on the quality and usefulness of predictions made by our algorithms.

OFFLINE EXPERIMENT

The offline experiment tests whether our algorithms are useful for predicting specific relevant citations for a given paper.

We started with a dataset based on the over 500,000 research papers available through ResearchIndex. In order to make that data more manageable for our algorithms, we chose to reduce this dataset by dropping papers that contained fewer than two citations and by dropping citations that were cited less then two times. The assumption when doing so was that these papers and citations do not contain enough ties to the rest of the dataset and thus only introduce noise in a CF framework. About 186,000 papers and 485,000 citations remained after removing these poorly connected papers. Papers that remained had an average of 16 connections to other items, either papers or citations, in the dataset.

Offline Experimental Design

This experiment consisted of removing citations from the test dataset and then attempting to predict those missing citations. For each paper in the test dataset, we randomly removed one citation from that paper's references list. The remaining citations were used to generate a list of recommended citations. We then found the rank of the removed citation in the list of recommendations. This protocol is similar to the "All but one" protocol in [1], but we also include coverage when calculating the prediction accuracy.

We divided our dataset into training and test datasets at a 90% to 10% ratio in the following manner: For 90% of the papers we wrote their citations into the training dataset. For the remaining 10% of the papers, we removed one citation to create a test case for the test dataset. We included the incomplete citation lists from the test cases in the training dataset.

We performed a 10-fold cross validation for this experiment, where in each fold we randomly assigned data into either the test or training datasets. Each recommendation algorithm was run with all of the datasets.

The experiment has some limitations. The most significant one was that our recommendation algorithms might discover citations that are more relevant than the one held out. Such citations may have not been included in the paper's references list because of limits on space or because they overlapped with other references, possibly the one left out. However, we feel that authors generally do a good job selecting citations for their papers, so we expected the removed citation to be recommended.

Another factor to consider was that our recommendation algorithms could recommend citations that did not exist when the paper in question was written. To deal with this problem, we filtered out recommendations with a publication year later than that of the target paper.

Offline Experiment Algorithms

Five of our six algorithms were used for the offline experiments. We did not include Google Baseline because it would have been an unfair and heavy use of Google's search engine.

Offline Experiment Metrics

We examine three metrics for this experiment: rank, coverage, and effective rank. We define rank as the position of the test citation in the filtered recommendation list. Rank is a proxy for user utility, since users prefer to find relevant results earlier. In this context, rank is also similar to a precision metric from information retrieval applications.

Coverage is the percentage of the time that an algorithm is able to successfully make a recommendation. Normally, this means that the algorithm was able to make *some* recommendations. In this experiment, we define a "successful" recommendation as one that includes the test citation. As a complement to rank, coverage is similar to a recall metric from information retrieval.

Since users are most interested in how much value they will get from an algorithm in practice, we calculate an *effective rank* metric that scales the recommendation percentage by coverage for each rank tested. When measuring the percentage of times a recommender returned an item below a certain rank, effective rank penalizes recommenders for those situations it was unable to recommend any item at all. Thus, the effective rank metric, as a combination of rank and coverage, is similar to the F1 or other combination metrics found in information retrieval.

Offline Experiment Results

Figure 1 shows the performance of each recommender at recommending the removed citation within its top 1, 10, 20, 30, or 40 recommendations. Item-Item, User-Item, and Graph Search all perform substantially better than either the Co-citation or Bayesian recommenders.

However, Figure 1 does not take coverage into account. The five recommenders vary greatly in their coverage. The Bayesian recommender can eventually recommend 100 percent of the removed citations. Localized Graph Search, on the other hand, only searches a small part of the citation graph, generating predictions for only 28.1% of these citations. Item-item (51.3%), Co-Citation (62.4%), and User-item (67.5%) fall between these two extremes.

Figure 2 reveals that taking coverage into account by using the effective rank metric described earlier greatly changes the relative performance of the recommenders. Graph Search falls to the bottom, while Naïve Bayesian improves to the middle of the pack. User-Item and Item-Item continue to lead all recommenders, with User-Item's greater coverage putting it above the rest. The differences between the User-Item and Item-Item recommenders versus the other three were statistically significant at all ranks.[2]

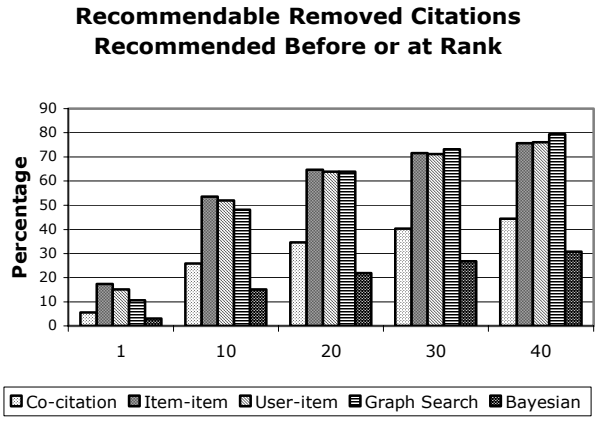

Figure 1: For removed citations that an algorithm was able to recommend, the percentage of citations recommended first, and in the top 10, 20, 30, or 40 by each algorithm

Offline Experiment Discussion

Graph Search performed well on the papers it was able to recommend for, but based on its coverage, it was "cherry-picking" easy-to-recommend test cases. This emphasizes the importance of the effective rank results. Recommendation algorithms must both provide high quality recommendations, and be able to provide them over as much of the dataset as possible.

Our test cases were slightly biased toward oft-cited papers, because papers that were cited more often were more likely to be test cases. Algorithms that are better at recommending papers that are cited many times had an advantage.

ONLINE EXPERIMENT

While the offline experiments allowed us to test the effectiveness of our algorithms at a coarse level, it could not assess the value of the recommended citations. What if an algorithm recommended better or equivalent citations? We felt that only an experiment with real people could provide us with that information. We used the offline experiment dataset for the online experiment as well.

Online Experimental Design

We created an online survey for authors of papers in ResearchIndex to rate the relevance of citations recommended for a paper they had written. We focused on authors because we wanted people who were intimately familiar with a paper to make the judgments about our recommendations. We did not attempt to verify that a subject in the study was actually the author they claimed to be, as we felt there was little incentive for people to lie.

[2] Unless noted otherwise, significance tests are at $p < 0.05$.

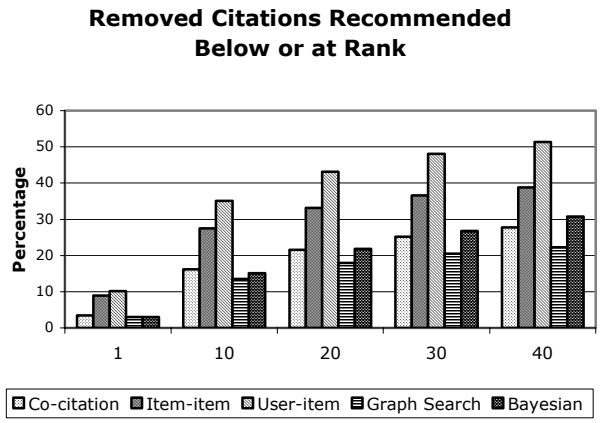

Figure 2: For all removed citations, the percentage of citations recommended first, and in the top 10, 20, 30, or 40 by each algorithm

Subjects were invited to our survey through a link from ResearchIndex. After consenting to participate, we asked each user for her name, as it would usually appear in a paper. We then searched for papers in our test dataset that the author had written and asked the author to choose a paper. Each author was randomly assigned to one of our six recommendation algorithms.

The algorithms generated a list of five recommended citations that were not written by the subject. We felt that recommending an author's own papers would not be helpful for this experiment since few authors forget to cite their own work.

Sometimes an algorithm could not produce a recommendation list for a given paper. If this happened, we dynamically reassigned the subject to other recommendation algorithms until we were able to generate recommendations. Data from these subjects is not included below, as we were afraid that an algorithm's inability to produce recommendations for a paper might indicate that it was a "hard" paper to recommend for.

The Survey

For each recommended citation, the subject answered the following two questions: "How relevant is this citation to your paper and its related work?" with the scale: Relevant and a good addition, Relevant but redundant, In the same field but not relevant, In a different field and not relevant, and No Answer, and "How familiar are you with this citation?" with the scale: I have cited this myself, I have read this but not cited it, I have heard of this but not read it, I do not know this at all, and No answer.

The survey provided a link to the subject's original paper and links to the ResearchIndex entry for each of the recommended citations so that authors could investigate citations that they were not familiar with.

After evaluating all five citations, we asked the subject about the citations as a whole. These questions were on a five-point scale of Excellent, Good, Fair, Poor, and Terrible. We asked about the overall quality, usefulness, and novelty of the recommendations. We also asked for an overall rating of the recommendations, "all things considered".

Finally, we asked two questions about whether the recommendations would be suitable for finding related work and generating reading lists. These questions used a scale consisting of Definitely Yes, Probably Yes, Maybe, Probably Not, and Definitely Not.

Subjects were also given an opportunity to provide extra comments they might have had about the experiment.

Table 1: Number of subjects for each recommender

Algorithm	Users
Co-Citation	23
User-Item CF	31
Item-Item CF	30
Naive Bayesian	19
Local Graph Search	26
Google	28

Online Experiment Algorithms

We used all six algorithms in our online experiment.

Online Experiment Metrics

The answers for each question were tallied and expressed as percentages. For some questions we merged answers from multiple categories into baskets. For example, a 'relevance' basket would include both 'relevant and a good addition' and 'relevant but redundant', but not include the other possible answers for that question.

Online Experiment Results

Table 1 shows the number of users who received recommendations from each recommendation algorithm. Table 2 shows the overall user opinion from the online experiments.

Figures 3 and 4 show users' judgment of the relevance and novelty of recommendations they received. Both figures use the basketing approach described in the metrics section. Since we excluded "no answer" responses in both figures, the percentages in a figure for a given recommender do not add up to 100 percent.

Figure 3 shows the percentage of recommendations users considered relevant and irrelevant for each recommender. We considered that a recommendation was relevant if the user chose either the "relevant and a good addition" or the "relevant but redundant" response. The Google, Naïve Bayesian, and Graph Search algorithms tend to do better than the other algorithms, with one in two citations judged as relevant. The lowest-rated algorithm, Item-Item, returns a relevant citation 25% of the time. Graph Search and Google dominate the three least relevant, and the difference between them and those three is statistically significant.

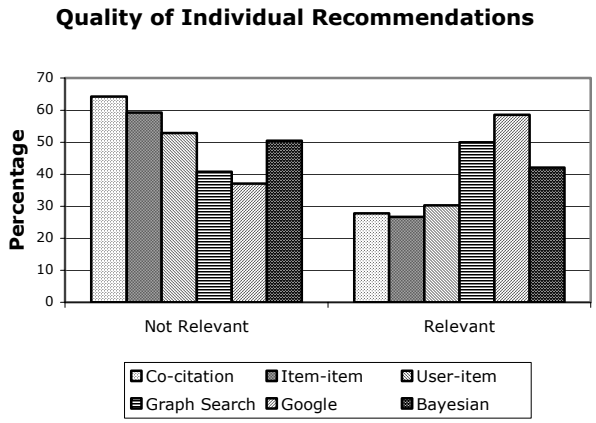

Figure 3: Users' judgment of the relevance of individual recommendations, grouped by algorithm

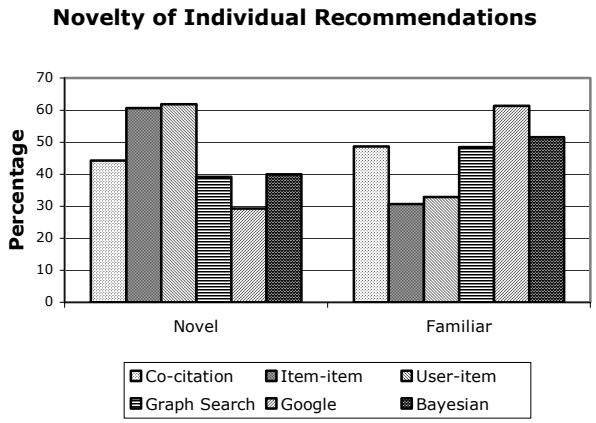

Figure 4: Users' judgment of the novelty of individual recommendations, grouped by algorithm

Figure 4 shows the percentage of recommendations users considered familiar and novel for each recommender. We considered a recommendation familiar if the user chose either the "I have cited this myself" or the "I have read this, but not cited it" response. The User-Item and Item-Item algorithms tend to produce more novel recommendations than the others. Google produced the least novel recommendations. The differences between Google and all of the other algorithms were statistically significant, as was the difference between both Item-Item and User-Item against all of the other algorithms.

Google produces both the most relevant and familiar citations, while Item-Item produces the least familiar and least relevant citations. We measured the correlation between a user's relevance and novelty ratings to be –0.51. This means that as novelty goes up, relevance tends to go down.

Most users rated the overall, quality, usefulness, and novelty of the recommendations they received in the middle of the range, with more "poor" (43%) than "good" (25%) ratings. The best quality was Graph Search, the best usefulness was Google, the best novelty was User-Item, and the best "overall" was Graph Search. On the questions about the helpfulness of the recommendations for research-related tasks, users also rated toward the middle of the range, but were more likely to say "probably yes" (50-56%) than "probably no" (22-32%).

Users' perception of the usefulness of the recommendations for research tasks differed based on the recommender that generated the user's recommendations. Figures 5 and 6 show results for the task-related questions broken down by recommendation algorithm. In both figures, we counted "Definitely Yes" and "Probably Yes" responses as helpful and "Definitely No" and "Probably No" responses as unhelpful. We did not count "Maybe" or blank responses.

Figure 5 shows the percentage of users who thought that a system which generated recommendations like those they received would be helpful for finding related work. The Graph Search algorithm, which produced the most relevant recommendations, was ranked most helpful for this task, and Google, the most likely to generate familiar and relevant citations was also considered very helpful.

Figure 6 shows the percentage of users who thought that their recommendations might be useful in finding papers to read. The Graph Search algorithm also performs well here, while the User-Item and Naïve Bayesian algorithms move up relative to other algorithms. The difference between both Graph Search and Google to Item-Item was statistically significant.

Online Experiment Discussion

More users thought that the algorithms would be helpful than not helpful, both for finding related work and for

Table 2: Overall User Opinion, in percentages

	Terrible	Poor	Fair	Good	Excellent
Overall Quality	11	29	31	26	3
Overall Usefulness	16	32	31	20	1
Overall Novelty	7	28	35	25	5
All Things Considered	12	32	32	25	1
	Definitely Not	**Probably Not**	**Maybe**	**Probably Yes**	**Definitely Yes**
Helpful Finding Related Work	8	24	18	34	17
Helpful Finding Papers to Read	8	15	21	38	19

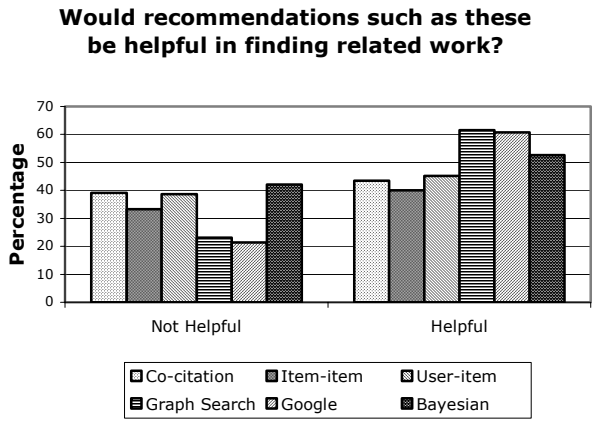

Figure 5: Users' judgment of the helpfulness of recommendations in finding related work, disregarding neutral and blank responses, grouped by algorithm

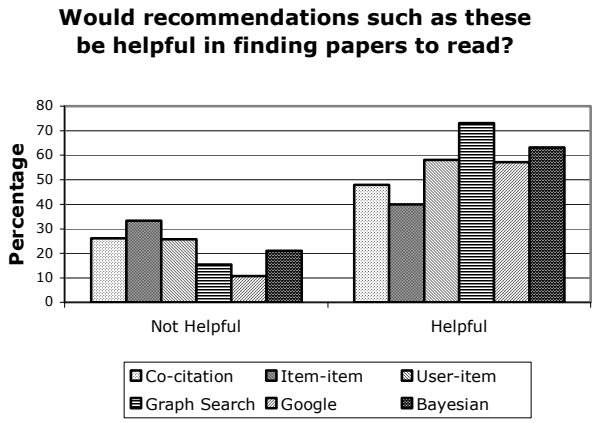

Figure 6: Users' judgment of the helpfulness of recommendations in finding papers to read, disregarding neutral and blank responses, grouped by algorithm

locating papers to read. This was true both on average over all algorithms and for each individual algorithm on every task.

This result seems at odds with users' judgments on the relevance and novelty of individual recommendations. However, consider that even the worst algorithm in terms of quality generated one relevant recommendation in four. Over 60% of users received one or more recommendations that they considered to be a good addition. Another 16% received at least one that was relevant but redundant.

All four CF-based algorithms were able to make recommendations for every paper. The baseline algorithms were not, with Graph Search failing on 19% and Google failing on 39% of recommendation requests. Scaling users' perception of the helpfulness of these algorithms by their ability to make recommendations makes Google fall to the bottom, and makes Graph Search roughly comparable to User-Item.

The importance of this "cherry-picking" behavior for the baseline algorithms, algorithms which are modeled after how people currently search for related work, cannot be stressed enough. It is important for two reasons. First, these algorithms work well only for the subset of the papers they could search over. This limitation over the coverage of the space greatly reduced the number of possible recommended citations. Second, both of these algorithms scored very high on familiarity. Neither of these algorithms could be effectively used to find novel recommendations.

In order to find both relevant and novel recommendations, no one single algorithm stands out. One interpretation is that an effective system might incorporate several algorithms. This could be done in several ways:

- Try the algorithms in descending order of quality until one produces recommendations.
- Make recommendations from all algorithms, and combine them, perhaps by learning which users like which algorithms and weigh the algorithms' recommendations.
- Use different recommenders for different tasks. CF-based algorithms produced more novel recommendations that might be more useful for finding reading lists, while the baseline algorithms might be more useful for finding related work.

The inverse correlation between novelty and quality was also interesting. Many novel recommendations are indeed not relevant. However, we received user comments stating that it was difficult to accurately judge relevance only from the information provided by a citation. This means that systems for exploring citation webs such as ResearchIndex must make it easy for users to quickly judge a citation's relevance by providing as much information as possible about novel citations.

DISCUSSION

The User-Item and Item-Item algorithms performed very well in the offline experiments, but did not do nearly as well at locating relevant citations in the online experiments. There are several possible reasons for this discrepancy. Since authors often cite themselves, the offline experiment would often test whether an algorithm could find other relevant papers by the same author. In the online experiment, we specifically prohibited algorithms from recommending citations by the author of the paper. Since these self-citations are presumably relevant and not novel, excluding them would tend to decrease relevance and increase novelty.

In many usage scenarios, it would be appropriate to recommend citations by the same author. We believe that the User-Item and Item-Item algorithms will perform better online in such scenarios. Further, the offline experiment constrained the recommenders to only recommend older

references, while the online experiments allowed all references. There may be properties of the citation graph that lead some algorithms to perform better for discovering older papers, while other algorithms do better at finding newer papers.

Different algorithms performed better along different dimensions of the tasks. User-Item and Item-Item were the dominant algorithms in the offline experiments by the effective rank measure. On the other hand, real users rated Item-Item weak at relevance, and User-Item was in the middle of the pack at relevance. Users much preferred User-Item for novelty, though, scoring Google, Graph Search and Bayesian as the least novel. In general, algorithms that performed well at relevance performed poorly at novelty.

Rather than identify one algorithm as the single best for all applications, our experiments lead us to believe that different usage scenarios would likely lead to different best algorithms. For instance, consider the application of finding a few references to complete a reference list for a nearly complete paper. This scenario was most like our online experiment. In general, our users preferred highly relevant references for this application, and were skeptical of the novel references suggested to them. One of our users said, "My paper's references were much more relevant (as to be expected)". There might be benefits to our community if users sought more novel references, but our surveys suggest that users would not use a recommender that pushed them in the direction of novelty. Therefore, the best recommenders for this application would be one that produced the most relevant recommendations for users, which was the Google recommender. Since it fails 2/5 of the time, falling back to the second most relevant algorithm, Naïve Bayesian, would be best in these cases.

A weekly reading group seeking interesting papers to expand their horizons might have very different goals. Google and Graph Search would tend to keep them within a narrow area of closely related papers. For instance, one user said of Graph Search, "I have cited all works recommended by your system in some paper. They are certainly very related. However, they lacked novelty in this case." This application would do better with an algorithm that scored high in novelty, such as User-Item, about which one user said, "They would be somewhat useful if I was trying to get a broader understanding but not if I was trying to get a better understanding of the problem my paper was solving." Thus, User-Item would be less suitable for finding closely related references, but perhaps more suitable for finding novel references for a reading group.

Some usage scenarios fall in between these two. A researcher entering a new research area in which he has read a small number of papers might like a recommender that can help him find a larger collection of papers that span the new area. Here, relevance is important to find papers that are within the new field. But, novelty is important too, since otherwise the papers may be too narrowly focused. We have observed this problem in earlier research in which the Item-Item algorithm tended to recommend movies that were very closely related to previously seen movies, resulting in too narrow a view of a user's tastes [14]. For these applications, the best recommenders are those that do well at both relevance and novelty. The User-Item algorithm was best at relevance in the offline experiment, fourth best at relevance in the online experiment, and best at novelty in the online experiment, so it should do well in this scenario. An even better solution might be to combine User-Item with Graph Search or Naïve Bayesian, the recommenders users judged more relevant, to make a hybrid algorithm that did well on both relevance and novelty.

CONCLUSION

Recommender systems have a rich future in recommending research papers. A recommender can be a great tool to help researchers find the papers that will be most relevant to them. Our experiments show that the choice of algorithm affects the type of recommendations produced. The best algorithms we studied can either provide very relevant recommendations or very novel recommendations, though we found no single algorithm that provided both at the same time. The use of the citation web to avoid the startup problems that are a challenge for collaborative filtering was effective. These algorithms all ran with very little data from the user: just a citation to one of his papers, which we were able to look up in ResearchIndex.

In the future, it would be interesting to see how other recommendation algorithms would perform for these experiments, especially text-based and hybrid algorithms [20]. It would also be interesting to see these algorithms used with citation webs of research papers from other fields, especially those in which the complete text of most published papers are available online.

One application domain that would be interesting would be to find people who are relevant to a paper. For instance, a recommender could generate a list of *people* rather than citations. An editor might use this recommender to locate good reviewers for a paper. Editors sometimes search for reviewers among the authors of the cited papers. Since these are often closer colleagues of the author, that list may not be broad enough. Using such a recommender would broaden the pool of potential reviewers.

Another idea comes from one of our users, who said: "Suppose I were working on a sequel (much research is, to some extent, a sequel of earlier work). I am vitally interested to know about LATER papers that either cite my earlier work (or should have or might have or is somehow closely related)". The recommenders mentioned here could easily be tuned to only produce later papers, rather than earlier papers as they did in the offline experiment.

While the two experiments were very different, and while the algorithms varied in behavior, we believe they show that recommenders based on the social web of citations in

published research papers can be valuable aids to researchers.

ACKNOWLEDGEMENTS

We would like to thank to Steve Lawrence, David Pennock and their colleagues at NEC Research for their wonderful cooperation in helping us both the ResearchIndex dataset for the offline experiments and with linking ResearchIndex to our Web site for the online experiment. We would also like to thank Ed H. Chi for his valuable comments. This work is supported by grants from the NSF (DGE 95-54517, IIS 96-13960, IIS 97-34442, IIS 99-78717, and IIS 01-02229) and by Net Perceptions, Inc.

REFERENCES

1. Breese, J., Heckerman, D., and Kadie, C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proc. UAI 98*, Madison, 1998, 43–52.
2. Bollacker, K., Lawrence, S., and Giles, C. L. Discovering relevant scientific literature on the web. *IEEE Intelligent Systems*, 15(2), 42–47, 2000.
3. Egghe, L., and Rousseau, R. *Introduction to Informetrics*. Elsevier, Amsterdam, 1990.
4. Friedman, N., Gieger, M., and Goldszmidt, M. Bayesian Network Classifiers. *Machine Learning*, 29, 131–163, 1997.
5. Garfield, E. *Citation Indexing: Its Theory and Application in Science, Technology, and Humanities*. Wiley, New York, 1979.
6. Goldberg, K., Roeder, T., Gupta, D., and Perkins, K. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval Journal*, 4(2), 133–151. 2001.
7. Herlocker, J., Konstan, J. A., Borchers, A., and Riedl, J. An Algorithmic Framework for Performing Collaborative Filtering. In *Proc. SIGIR 99*, Berkeley, 1999, 230–237.
8. Hill, W., Stead, L., Rosenstein, M. and Furnas, G. Recommending and evaluating choices in a virtual community of use. In *Proc. CHI 1995*, Denver, 1995, 194–201.
9. Karypis, G. SUGGEST Top-N Recommendation Engine. Available for download from http://www.cs.umn.edu/~karpyis/suggest/.
10. Kautz, H., Selman, B., and Shah, M. Referral Web: Combining Social Networks and Collaborative Filtering. *Communications of the ACM*, 40(3), 63–65, 1997.
11. Lawrence, S., Bollacker, K., and Giles, C. L. Indexing and Retrieval of Scientific Literature. In *Proc. CIKM 99*, Kansas City, 1998, 139–146.
12. Lawrence, S., Giles, C. L., and Bollacker, K. Digital libraries and autonomous citation indexing. *IEEE Computer*, *32*(6), 67–71, 1999.
13. Newman, M. E. J. Scientific collaboration networks: I. Network construction and fundamental results. *Phys. Rev. E 64*, 016131, 2001.
14. Rashid, A. M., Albert, I., Cosley, D., Lam, S. K., McNee, S. M., Konstan, J. A., and Riedl, J. Getting to Know You: Learning New User Preferences in Recommender Systems. In *Proc. IUI 02*, San Francisco, 2002, 127–134.
15. Resnick, P., Iacovou, N., Sushak, M., Bergstrom, P., and Riedl, J. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proc. CSCW 94*, Chapel Hill, 1994, 175–186.
16. Sarwar, B., Karypis, G, Konstan, J. A., and Riedl, J. Item-based Collaborative Filtering Recommendation Algorithms. In *Proc. WWW 10*, Hong Kong, 2001, 285–295.
17. Scott, J. *Social Network Analysis: A Handbook*, 2nd Edition. Sage Publications, London, 2000.
18. Shardanand, U., and Maes, P. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proc. CHI 95*, Denver, 1995, 210–217.
19. Terveen, L., Hill, W., Amento, B., McDonald, D., and Creter, J. PHOAKS: A system for sharing recommendations. *Communications of the ACM*, 40(3), 59–62, 1997.
20. Woodruff, A., Gossweiler, R., Pitkow, J., Chi, E.H., and Card, S. K. Enhancing a Digital Book with a Reading Recommender. In *Proc. CHI 2000*, Amsterdam, 2000, 153–160.