Automatically Building Research Reading Lists

Michael D. Ekstrand[1], Praveen Kannan[1], James A. Stemper[2], John T. Butler[2], Joseph A. Konstan[1], John T. Riedl[1]

[1]GroupLens Research
Department of Computer Science and Engineering
University of Minnesota
{ekstrand,kannan,konstan,riedl}@cs.umn.edu

[2]University of Minnesota Libraries
{stemp003,j-butl}@umn.edu

ABSTRACT

All new researchers face the daunting task of familiarizing themselves with the existing body of research literature in their respective fields. Recommender algorithms could aid in preparing these lists, but most current algorithms do not understand how to rate the importance of a paper within the literature, which might limit their effectiveness in this domain. We explore several methods for augmenting existing collaborative and content-based filtering algorithms with measures of the influence of a paper within the web of citations. We measure influence using well-known algorithms, such as HITS and PageRank, for measuring a node's importance in a graph. Among these augmentation methods is a novel method for using importance scores to influence collaborative filtering. We present a task-centered evaluation, including both an offline analysis and a user study, of the performance of the algorithms. Results from these studies indicate that collaborative filtering outperforms content-based approaches for generating introductory reading lists.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering, retrieval models, search process*; H.3.7 [**Information Storage and Retrieval**]: Digital Libraries—*systems issues, user issues*

General Terms

Algorithms, Experimentation, Human Factors

Keywords

citation web, collaborative filtering, digital libraries, recommender systems, user study

1. INTRODUCTION

Researchers new to a field — both new researchers, such as first-year Ph.D students, and experienced researchers transitioning to work in a new field — face a challenge in "reading in" to that field. They need to be able to locate the seminal papers that initiated inquiry into that domain, and to be confident that what they are reading gives them a sufficiently complete understanding of the topic that they can map out the existing literature.

Researchers currently accomplish these tasks by following reference lists in already-identified papers and surveys, searching literature collections (commonly using tools such as Google Scholar), and receiving reading recommendations from researchers already working on the topic. This type of knowledge exploration seems like a good candidate for enhancement with computer-based tools. Looking at a reference list or set of search results, it is difficult for a researcher unfamiliar with the field in question to evaluate which papers have been influential in the development of the field and identify the fountainheads of streams of research. If algorithms could help filter through the literature and bring to light those papers which have had high impact, they could make the task of familiarizing oneself with existing work significantly easier. Based on previous successes in applying recommender technology to problems in the research paper domain [14, 22], we present here our investigation into methods of automating the production of reading lists.

To further refine our notion of the user's task, we assume that a user has seen or been given a few papers on the topic they want to read about, but does not know what the important papers in that field are. Therefore, the task considered in this work (which we call the Reading List task) is to take a set of papers, called the *query set*, and produce a short reading list that will serve as a good starting point for understanding the field of which the query set is a part. Following human-recommender interaction theory [16], we used the Reading List task to drive the design of both the algorithms and their evaluation. Task-centered evaluations allow an algorithm to be optimized for its utility to a particular task rather than generic prediction accuracy on a ratings set [15]. This has the potential to result in systems that are better able to meet users' particular needs.

The primary technical aspect of this work is a presentation and evaluation of methods for using known structure in the item space to improve recommendation quality. Traditional collaborative filtering (CF) approaches are "blind" — they have no knowledge of the items or of any relationships that may be intrinsic in the domain. The method we use for applying CF to the research literature gives it some knowledge of relationships, but this knowledge is local, extending only to co-citations [14]. Content-based approaches have knowledge of various characteristics of the items themselves,

http://doi.acm.org/10.1145/1864708.1864740
RecSys2010, September 26–30, 2010, Barcelona, Spain.

but many do not take into account item relationships. Link ranking algorithms, such as PageRank, HITS, and SALSA, attempt to computationally measure the global importance of individual nodes in directed graphs. The exact human-perceivable characteristic captured by these algorithms is unknown (and may vary from algorithm to algorithm), but they have proven effective as a means of measuring the importance and/or influence of resources in citation or link graphs. Throughout the rest of this paper, we will use "importance" to mean the combination of a paper's importance as a stand-alone work and its influence on subsequent work captured by the link ranking algorithms.

We test several approaches for incorporating importance metrics into existing recommender approaches to determine whether they enable the resulting system to create better reading lists. Among these approaches is a novel method for using augmenting collaborative filtering with these global graph ranking scores.

Our work, and this paper, is divided into three main components. First, we build a diverse pool of candidate recommender algorithms with and without link structure information (a total of 177 algorithms in 5 families). We then perform an offline evaluation of the candidates to select the best algorithms to represent the various families under consideration. Finally, we present the results of a user study in which we asked researchers to evaluate the quality of reading lists produced by the representative algorithms selected by offline analysis.

2. RELATED WORK

Recommender systems, particularly ones based on collaborative filtering, have been widely studied for some time. They have been applied to a variety of domains, including USENET articles [20], jokes [6], college courses [9], and research papers. In applying collaborative filtering to research papers, there are two major approaches. The most straightforward, exemplified by CiteULike [1], is to treat users' reference collections as their histories or purchase baskets and provide recommendations based on similarities in reference collections. An alternative approach, which avoids the cold-start problem of obtaining sufficient user profiles to generate recommendations, is to use the citation web itself to compute similarities and thereby generate recommendations. This is the approach taken by TechLens [14, 22] and forms the basis of our application of CF.

A variety of algorithms have been proposed for computing the importance of nodes on a directed graph. Many of these were initially conceived in the context of web pages connected by hyperlinks, but they are readily applicable to other directed graph contexts such as the web of citations in research literature. Of the available algorithms, we consider three. PageRank uses a random walk to rate the quality or authority of web pages based on the assumption that high-quality, authoritative pages will be linked to frequently, particularly by other good pages, and thus be well-connected [3, 19]. HITS uses mutually-reinforcing concepts of "hubs" and "authorities" to similarly rate web pages [8]. SALSA strikes a middle ground, using a stochastic approach on a bipartite hub/authority graph to avoid the topical tunnel vision sometimes exhibited by HITS [11].

Several authors have extended these algorithms to compute importance relative to a fixed set of known authorities, a natural mapping for the reading list task. White and Smyth present several such algorithms, some based on HITS or PageRank and some independently designed [23]. Chang et al. proposed another method in the context of creating customized authority lists [5], but the gradient descent used in their algorithm significantly decreases the sparsity of the link graph matrix, causing tractability problems for this approach.

Ranking algorithms are frequently used in combination with text-based information retrieval methods [8], which themselves constitute a form of content-based recommender system. We are not aware of any prior work combining these ranking algorithms with collaborative filtering. Massa and Avesani's method for integrating trust weightings into user-user collaborative filtering is somewhat similar [13]; while their method can be described as weighted user-user CF, we propose user-weighted item-item CF.

Our CF approach is also similar to CC-IDF [10] and the authority vector approach of [12]. Both of these computations are effectively weighted versions of bibliographic coupling, and are aimed at computing node similarity rather than authority (although the authority vector method does use local authority to influence similarity); our weighted CF uses global authority and is more closely related to co-citation than coupling.

3. ALGORITHMS

We considered a variety of algorithms for recommending reading lists. When combined, they yielded a pool of 177 recommender algorithms, plus additional algorithms eliminated early in our testing. These algorithms are assembled from various pieces; figure 1 shows the structures of the different classes of algorithms we considered.

3.1 Graph Ranking

To compute the importance of a paper based on its connectivity in the citation web, we considered three core algorithms. These algorithms all fit the same basic structure: given a directed graph $G = \langle V, E \rangle$, they produce a scoring function $r : V \to \mathbb{R}$ such that $r(a)$ is the measure of the importance of node a in the graph.

PageRank [3, 19], the heart of the Google search engine, scores nodes with their probabilities in the stationary distribution of a modified random walk on the link graph. This is a well-known ranking algorithm, and the underlying random process is intuitive and well-suited to modeling browsing and reading behavior. The resulting scores represent the probability of viewing a particular node through browsing the citation or link web; important papers should be reachable through many paths and thus have high scores.

HITS [8] produces two rankings of the graph. The *authority* score rates a node's authority, as measured by the extent to which it is referenced by other work, and the *hub* score rates a node's ability to point to authoritative sources. These relationships are mutually reinforcing (good hubs reference good authorities), and the scores are computed as the fixed point of this relationship. HITS is also well-known, and its distinction between a site or paper's role as a hub and as an authority is particularly appealing for trying to find important, authoritative papers. We used the HITS authority score for our ranking.

SALSA [11] is a stochastic algorithm similar to HITS, but is based on a random walk on the bipartite hub-authority network. It represents a middle ground between PageR-

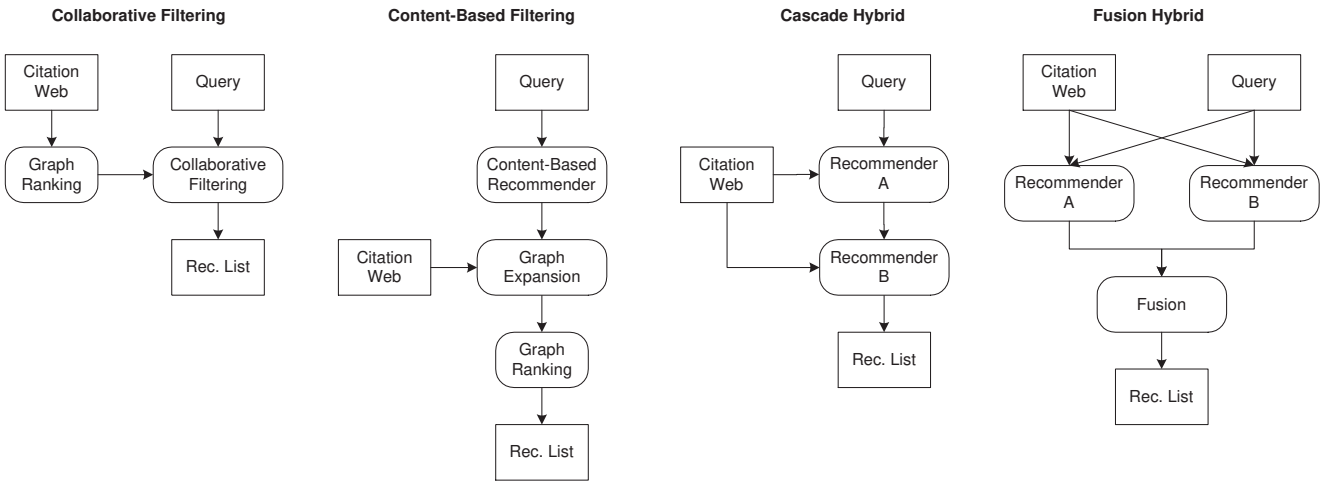

Figure 1: Recommender algorithm structures

ank and HITS, providing a stochastic method with distinct hub and authority ratings. Further, the authors argue that SALSA is able to pick out the authorities of several topics in a multi-topic graph, whereas HITS will likely only identify the authorities on a single topic. This ability could give it an edge in dealing with graphs that span several well-defined topics in the literature as well as providing more useful results when applied to the entire web. As with HITS, we used the resulting authority scores.

We also considered two biased ranking algorithms. In their original, forms, HITS, PageRank, and SALSA all compute an absolute measure of importance in either the complete graph or a topical subgraph, not considering any additional information that may be known about node importance. The biased algorithms compute importance relative to a particular set $S \subset V$ (in our case, the user's query set) of nodes. That is, they take S to be a set of nodes known to be important and use their links to influence the importance scores of the rest of the network. The insight for trying these algorithms as well is that they may be better at identifying the important papers most relevant to the user's query.

The first of the biased algorithms is HITS with Priors, which interprets HITS stochastically and extends it with a prior that causes the nodes in S to be preferred. The other is k-step Markov importance, a stochastic algorithm based on fixed k-step random walks originating at nodes in S; we use $k = 10$. Both of these algorithms are described by White and Smyth [23].

In all cases, our network is built from the reference lists of the articles in our data set: there is an edge from a_1 to a_2 if a_1 cites a_2. We also normalize importance scores to have a maximum score of 1 (rather than summing to 1, as many of them naturally do).

3.2 Collaborative Filtering

Our collaborative filtering algorithm is standard item-item collaborative filtering in a unary ratings space [7]. This algorithm is widely used, has proven to perform well in a variety of applications, and the implications of implementing it over a unary item space are well-understood. We build the ratings matrix from the citation web by treating each article as both a user and an item, where each article has "purchased" the articles it cites [14]. This allows collaborative filtering to be done using information available in the article metadata, avoiding the user cold-start problem that plagues many applications of collaborative filtering.

We tested three versions of the basic CF algorithm. The first, *cf-plain* is raw item-item CF without any normalizations applied. The second, *cf-unit*, uses both of the normalizations proposed by Karypis [7]: in an e-commerce domain, the user purchased vectors are normalized to unit vectors prior to similarity computation and item-item similarity vectors are likewise scaled to unit vectors. For the user vector normalization, if $\mathbf{u}$ is the vector of user u's purchases such that u_i is 1 if u purchased i and 0 otherwise, the normalized vector $\hat{\mathbf{u}} = \mathbf{u}/\|\mathbf{u}\|_2$. This normalization has the effect of causing users who have purchased fewer items to exert more influence on the similarity of the items they have purchased. The item similarity vector normalization operates similarly on the rows of the item-item similarity matrix and has the effect of increasing the influence of items with sparse neighborhoods (and thus higher information value in their neighborhoods) on the final predictions. The third CF variant is *cf-full-unit*, a variant of *cf-unit* which uses the total reference list length (*cf-unit* only counts references to other papers in the data set we used when computing the citation vector length; details on the data set are in Section 4.1).

To use graph ranking to influence collaborative filtering, we weight papers by their graph importance score prior to building the item-item similarity matrix. Our approach was inspired by Karypis's user vector normalization. To integrate a graph ranking algorithm, we replace the user-unit-vector normalization with a normalization step which multiplies each paper's citation vector with the paper's importance score $r(u)$ such that $\hat{\mathbf{u}} = r(u)\mathbf{u}$. This causes papers with higher importance scores (e.g. higher PageRank) to exert more influence on the similarity of papers they cite, thus biasing the collaborative filter to favor the "opinions" of more authoritative papers. We tried each of PageRank, HITS, and SALSA for weighting the collaborative filtering algorithm. Biased ranking algorithms are not practical in this context because they would require recomputing the item-item similarity matrix for each query.

3.3 Content-Based Filtering

The content-based algorithms we used locate similar papers by text matching on the title, abstract, and keywords provided by the paper. To find papers matching the query set, we concatenate the text of each of the papers in the query set into a single document and score papers based on their similarity to this document. In our previous work, this was considered as the CBF-Combined algorithm [22].[1]

We used the Lemur toolkit [18] (version 4.11) as our text search implementation. Our querying used Lemur's BM25 TF retrieval method with $k_1 = 1.2, b = 0.75$ for documents and $k_1 = 1000, b = 0$ for queries. These values are recommended by the Lemur developers based on TREC performance results [24, 21].

We tested two approaches for augmenting CBF with article importance scores: *subgraph ranking* and *blending*.

Subgraph ranking is the method used by Kleinberg for applying HITS to web search [8]: we compute the top 200 matches for the query, not including members of the query set itself, using Lemur, and take the union of these results and the query set to produce a base set B. We then add to B all papers citing or cited by papers in B. The members of B are combined with the citations between them to form a subgraph G' of the citation web, and are ranked by nonincreasing importance score within G'. We used all five ranking algorithms for this method.

The blending method scores each candidate paper using a linear combination of the search score reported by Lemur and its graph rank within the whole citation web, using the following formula (where $L(a)$ is the text search score and $r(a)$ the rank):

$$s(a) = \alpha L(a) + \beta r(a)$$

We determined the coefficients α and β by training a multivariate logistic regression against a training set of articles; the details of the training are described in section 4.2. We excluded the relative ranking methods from this method to avoid re-ranking the entire citation web for each query.

3.4 Hybridization Techniques

We tested three hybridizations of these algorithms: CBF-CF, CF-CBF, and Fusion [22]. CBF-CF takes the output of a CBF recommender, possibly with ranking, and uses it (along with the original query set) as the query for a CF recommender. CF-CBF does the reverse: the output of a CF recommender is unioned with the query set and used as the input to CBF. In Burke's taxonomy [4], these are both cascade hybrids.

Fusion, a weighted hybrid in Burke's taxonomy, runs a CF and a CBF recommender in parallel and blends the resulting ranked lists. The first items on the combined recommendation list are those items which appeared on both input lists, ordered by the sum of their ranks on each of the single-algorithm lists. After all items listed by both recommenders, the combined list contains the remaining items, alternating between the recommenders. Figure 2 shows an example of this process.

We applied each hybridization technique to each pairwise combination of CF and CBF algorithms. The breakdown of the resulting pool of 177 algorithms is shown in Table 1.

[1]We also tried CBF-Separated, but its computation cost and poor performance in early trials led us to abandon it.

	1	2	3	4	5	
List 1:	A	B	C	D	E	**Combined ranks:** A=5, B=3, E=8
List 2:	B	F	E	A	G	**List 1 remainder:** C, D
Combined list:						**List 2 remainder:** F, G
B	A	E	C	F	D	G

Figure 2: Example of Fusion on two lists

Class	Count	Notes
CF	6	raw, 2 unit-norm, 3 rank-weighted
CBF	9	plain, 3 blended, 5 subgraph-ranked
CBF-CF	54	each CBF as input to each CF
CF-CBF	54	each CF as input to each CBF
Fusion	54	each CF fused with each CBF

Table 1: Algorithms by category

4. OFFLINE EVALUATION

We performed an offline evaluation of our pool of recommender algorithms to understand their relative strengths and weaknesses. This also served as way to select a small number of algorithms to subject to human testing. This evaluation was intended to provide a reasonable simulation of the reading list task.

4.1 Data Set and Evaluation Strategy

Our data set, for both the offline evaluation and live user study, was a dump of the metadata in the ACM Digital Library from April 2010. The dump contains bibliographic information for 256,937 articles and nonempty reference lists for 201,145 of those (empty reference lists can be a result either of the paper lacking a reference list, or the digital library not successfully extracting the reference list from the hard copy or PDF). References to articles not in our article set were ignored for our citation web computations.

Previous work on research paper recommendation [14, 22] evaluated recommenders for the task of finding additional citations for in-progress papers by doing a hold-out test on randomly-selected papers in the literature: for each test paper, one paper was held out from its reference list and looked for in set of recommended papers. This strategy produced effective task-based evaluations for the problem of finding additional citations. In order to perform a similarly task-based offline analysis specifically geared towards evaluating performance in the context of building reading lists, we needed a set of good introductory reading lists. We used survey articles for this purpose. We expect survey articles to have bibliographies tailored to familiarizing readers with the history and current state-of-the-art of a topic; their reference lists, therefore, should provide a closer approximation of the reading lists we desire to produce than general research papers. To obtain a set of such lists, we used the reference lists of articles appearing in *ACM Computing Surveys* (CSUR) for our offline evaluation.

After removing all CSUR articles from our data set, each CSUR article with at least 15 references to articles remaining in the data set was used as a simulated reading list. This threshold ensured that test lists were long enough for the recommenders to work and had the side effect of excluding short research papers published as surveys. It also excluded papers referencing mainly papers outside the data set or

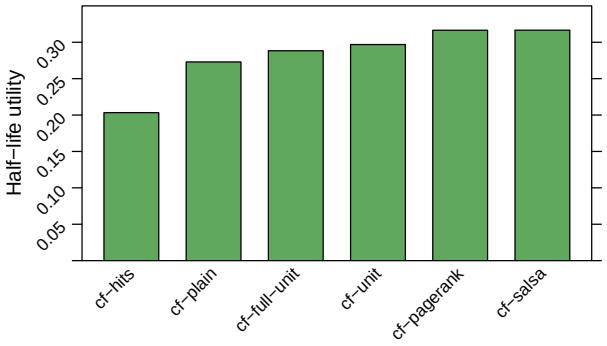

Figure 3: Collaborative filtering performance

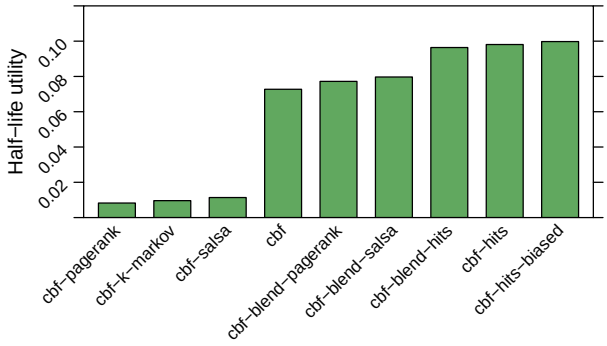

Figure 4: CBF performance

whose reference lists were unavailable. 220 out of 1185 total CSUR articles qualified as test lists. These articles were then split into two sets of 110 articles each, one to train the linear combination for blending ranks and text searches and the other to evaluate algorithm performance.

We performed the actual testing by randomly holding out 5 citations from each qualified list and using the remaining citations (at least 10) as the query set. Each recommender produced a ranked list of up to 100 papers. We evaluated the algorithms using the half-life utility metric [2] with a half-life α of 5 (the length of our target recommendation lists) and a utility $u_{a,i}$ equal to 1 if the paper at position i in the list of recommended papers is in article a's held-out citations and 0 otherwise:

$$R_a = \sum_i \frac{u_{a,i}}{2^{(i-1)/(\alpha-1)}}$$

Since all reading lists had the same number of held-out citations, the maximum utility $R^{\max} = 3.6426$ is the same for all articles, and a each recommender's performance R over the whole test set T is the fraction of total maximum utility achieved:

$$R = \frac{\sum_{a \in T} R_a}{|T| R^{\max}}$$

The resulting score is in the range $[0, 1]$, with a recommender that always placed the held-out items at the beginning of its list receiving a score of 1.

4.2 Training the Linear Blend

We trained the linear blend of content-based filtering and graph ranking using the same basic setup. We held out 5 items from each list in the training set and used the text of the remaining items as the query to Lemur. We learned a logistic regression for a response variable equal to 1 for articles in the target set and 0 otherwise against the text-similarity score and rank, with a separate regression for each ranking algorithm. The coefficients from this regression became the coefficients in the blending function; since our evaluation depends only on rank and not actual scores, the intercept and log function were irrelevant.

4.3 Individual Algorithm Results

Figure 3 shows the results for the collaborative filtering algorithms in our pool. CF with PageRank and SALSA performed better than standard collaborative filtering (*cf-unit*), and this difference is statistically significant ($p < 0.02$ in a matched-pairs t-test).[2] There was no difference between PageRank and SALSA weighting, and HITS weighting did not perform well.

Figure 4 shows the results achieved by the content-based algorithms. HITS ranking shows a slight improvement over other algorithms, but differences among algorithms performing at least as well as CBF are not significant (using ANOVA with Tukey's Honest Significant Differences method for multiple pairwise comparisons). It is interesting to note that PageRank and SALSA performed poorly in the subgraph-ranking configuration. These algorithms' performance picks back up in the linear blending configuration, suggesting that, unlike HITS, these algorithms may perform better when operating on the entire citation web than on a topical subgraph. For PageRank, this is not surprising, as PageRank is generally computed over the entire Web. SALSA is presented, however, using the subgraph-ranking configuration of HITS; our results suggest that SALSA may provide more benefit when applied to a graph as a whole rather than to a topical subgraph.

4.4 Overall Results

Figure 5 shows the results from the three best algorithms in each class (CF, CBF, and each of the hybridization strategies). Of the hybrids, CBF-CF performs the best, although not as well as plain collaborative filtering.

5. USER-BASED EVALUATION

While the offline analysis is useful in narrowing the set of candidate algorithms and providing preliminary insight into how the various algorithms will perform, user testing is required to assess actual performance for the task. To that end, we asked graduate students in the University of Minnesota Computer Science department to use three of the algorithms and evaluate the reading lists they provided.

Based on the offline analysis, we selected three representative algorithms for user testing:

- CBF with Biased HITS subgraph ranking (CBF)
- CBF fed through PageRank-weighted CF (CBF-CF)
- PageRank-weighted CF (CF)

[2]While we perform significance testing using ANOVA with Tukey's Honest Significant Differences generally in this paper, a matched-pairs t-test is appropriate for measuring a specifically-targeted single difference.

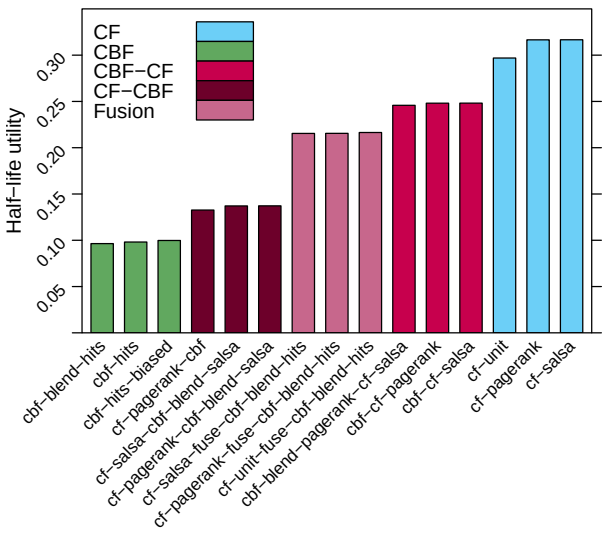

Figure 5: Performance of selected algorithms

We chose PageRank over SALSA for our CF weighting because it is used more widely and the two algorithms performed equivalently.

5.1 Methodology

Students using our survey tool first used a search form to construct a query set consisting of 5-10 papers. The tool then used their query set to build a 5-item reading list with each of the three recommenders and asked the user to evaluate each list on its suitability for introducing a new researcher to their topic and its coverage of the necessary aspects of the topic.

Assessing the coverage of a retrieval or recommendation system is difficult when experts are available; that difficulty is exacerbated when the target audience has less experience with the target domain. Many of the students we asked to participate had recently completed literature surveys, however, so there was at least one topic with which they should be fairly familiar. We attempted to get at coverage by asking how many of the articles on the recommended list they would keep in a 5-item reading list, assuming that the items discarded would be replaced with other articles filling in gaps in the recommended reading. Our attempt to evaluate coverage is limited, but can still provide some insight into how the reading lists were received.

After evaluating the recommendation lists, users were presented with a single list of all articles recommended by any of the tree algorithms in a and asked to evaluate each one on its relevance to their topic, importance within their topic, and their familiarity with the paper. Relevance was rated on a 1-5 scale, with 1 being "Exactly on-topic" and 5 being "Irrelevant"; importance on whether a reading list would need to contain 5, 10, 25, 50, or more items before they would include the paper (1-5 respectively). For further clarification, the tool picked one each of their highest- and lowest-rated papers on the relevance and importance scales (up to 4 papers total) and asked for a free-text explanation of why they rated it the way they did on that scale.

The final step of the survey asked users to order the three recommendation lists by quality and provide general feedback on the system, their assessment of its usefulness, and their stage of graduate education.

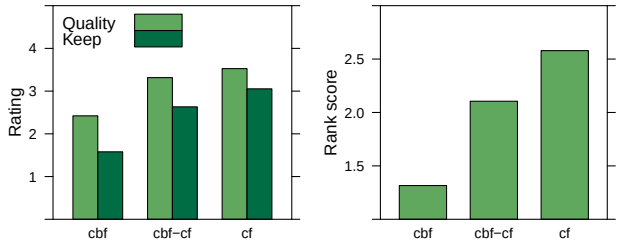

Figure 6: User evaluations of recommendation lists

	CBF-CF	CBF
CF	1.947	1.105
CBF-CF		2.368

Table 2: Average common items on reading lists between each pair of algorithms

5.2 Results

19 users completed our survey. Of these, 1 had already completed their Ph.D, 6 were M.S. students, and the remaining 12 were at various stages in pursuing a Ph.D. Unless otherwise mentioned, all significance tests in this section were performed using a block-design ANOVA with Tukey's Honest Significant Differences [17].

The user evaluations of recommendation lists as a whole are shown in Figure 6. The rank score is determined by giving a recommender 3 points for a 1^{st}-place ranking, 2 for 2^{nd}, and 1 for 3^{rd}. In general, users preferred the lists generated with at least some use of collaborative filtering. On both the quality rating and list ranking, CBF-CF and CF both had significantly better scores than CBF ($p = 0.01$). While the relative ranking of CF and CBF-CF was not significant in our responses, it is consistent with the offline results. As seen in Table 2, the recommendation lists also showed significant variation, particularly between CF and CBF.

Some users provided comments on some of the lists they were provided. Of particular note were comments we received on the diversity or lack thereof of the lists. One user, commenting on a list provided by CBF-CF, said the list was on a broad topic rather than the specific sub-area they were interested in. Another received a list (from CBF) that was too focused on a particular topic when they were looking for a broad list. CF gave another user a list that focused on a different but related topic; this is consistent with CF's high dependence on co-citations. These diverse experiences show that there remains work to be done on improving and measuring the consistency of providing recommendations in line with the desired topic scope.

Figure 7 shows the evaluations of the individual recommended papers by recommender. It shows the same ordering as was seen in the evaluation of entire lists on both relevance and importance. Some of the differences in relevance scores are statistically significant; CF produced more relevant results than CBF ($p < 0.01$). The importance and familiarity differences were not statistically significant (importance was marginally significant, with the ANOVA yielding $p = 0.06$

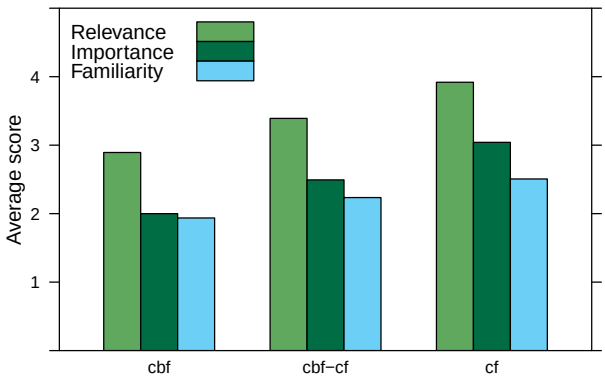

Figure 7: Mean user evaluation of individual recommended papers

for an effect in response to the recommender algorithm. CF vs. CBF was the significant pairwise difference).[3]

5.3 User Reception

The concept of the tool was well-received, with 13 of the 19 respondents either agreeing or strongly agreeing that "a tool like this would be useful for exploring unfamiliar areas of the research literature." Users also expressed specific appreciation for the idea, with comments like "Literature searches are difficult; any sort of automation would be greatly appreciated." and "With enough good data, this would be great."

Some respondents also provided us with specific shortcomings or useful features of the algorithms. One said:

> In my case, being interested in a specific problem doesn't mean I'm interested in a broader class of problems of which mine is a member. The tool is able to identify the broader classes, but it fails to discriminate which approaches are suitable for a specific problem.

Another was pleased with some distinctions our algorithms made:

> I asked about a difficult domain: social Q&A. It's a young research area, and it often gets confused with the field of NLP "QA" research. Your tool was able to disambiguate cleanly between research on social Q&A and NLP QA.

The positive response from users shows that systems of the kind we describe, if refined and made generally available, could provide significant benefit to the research community.

6. CONCLUSIONS AND FUTURE WORK

We have presented and empirically tested a large collection of recommender algorithms, showing that collaborative filtering seems to perform well for the task of generating reading lists for new researchers. This type of task-based evaluation is crucial as recommender systems are applied in more contexts. Using specific user tasks to drive algorithm design and evaluation strategy allows recommenders to be tailored to the specific needs of their target audiences and facilitates more accurate assessment of their true utility.

The core innovation of many of these algorithms was the use of graph ranking algorithms as an estimate of the importance or influence of a paper within the citation web. While the integration of graph ranking and text-based search is well-known, we have described a method of using graph rank weights to influence item-based collaborative filtering and shown in offline tests that it can improve accuracy of recommendations. This method is generally applicable to any collaborative filtering application where users can be weighted or scored on some criterion. In offline analysis, weighting the recommender provided a measurable boost in performance. The resulting recommender also performed well when assessed by users.

We were somewhat surprised to see collaborative filtering beat the hybrid approaches, as hybrid recommenders performed well in previous TechLens studies [22]. It seems likely that this is a result of CF having characteristics better-suited to the reading list task. It is also possible that this is a result of using item-item CF rather than the user-user algorithm in the previous work, but the results of a previous study showing user-user and item-item CF algorithms performing similarly in the research paper domain suggest that task suitability is a more likely explanation [14]. Since the CBF algorithms with ranking are similar to how search tools such as Google Scholar work, these results also suggest that collaborative filtering may be more useful than search engines for building introductory reading lists, although more specific and targeted research is needed before firm conclusions can be drawn on their relative merits.

The results from our survey indicate that recommender systems are a promising tool for helping new researchers acclimate themselves to the body of research literature in their fields. Our users responded well to the tool, with many of them indicating that they would find a generally-available tool based on these concepts to be useful in their research. The recommendation evaluations suggest that collaborative filtering is a good method for building reading lists, although more extensive user testing is needed to validate the offline results and preliminary user-based results we have presented. User feedback also showed that more work is needed on accurately assessing and operating within the scope of a user's query when providing recommendations.

Our algorithms omitted the user's query set from result lists. They are therefore incapable of telling the user that one of their query papers is a key paper to read on the topic. In real use where users may have some papers but not know their importance, this is a notable shortcoming. How can those papers be identified without just recommending back the query set? Further, what kind of user interface is needed to allow the user to make sense of such results? These questions do not have obvious answers, and resolving them would likely strengthen the benefit recommender-based tools provide to researchers.

It can be seen by comparing our results with the prior work that the specific task the recommender is asked to perform affects the relative performance for various algorithms. More work is needed to refine the present algorithms for gen-

[3]To control for the lack of independence in the individual article evaluations due to the articles being selected 5 at a time by each recommender, we computed the mean of each evaluation for each (user,recommender) pair and performed a block-design ANOVA across the means. The means constitute individual independent observations, satisfying the independence assumption of ANOVA.

erating reading lists, and similar task-driven efforts for other applications and domains will improve our understanding of the performance of recommender algorithms in specific contexts and the adjustments needed to provide maximum benefit to actual users of particular systems.

7. ACKNOWLEDGEMENTS

We gratefully acknowledge the support and assistance of our colleagues in GroupLens Research, and particularly Rich Davies for his assistance in developing and deploying the survey tool and Nandhini Raman for her contributions to the TechLens project. We also thank the anonymous reviewers for their useful feedback. This work is funded by the National Science Foundation under grants IIS 05-34939 and IIS 08-08692. Our data set was generously provided by the ACM Digital Library.

8. REFERENCES

[1] T. Bogers and A. van den Bosch. Recommending scientific articles using citeulike. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 287–290, Lausanne, Switzerland, 2008. ACM.

[2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43—52, 1998.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, Apr. 1998.

[4] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Nov. 2002.

[5] H. Chang, D. Cohn, and A. McCallum. Learning to create customized authority lists. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 127–134. Morgan Kaufmann Publishers Inc., 2000.

[6] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, July 2001.

[7] G. Karypis. Evaluation of Item-Based Top-N recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 247–254, Atlanta, Georgia, USA, 2001. ACM.

[8] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[9] G. Koutrika, B. Bercovitz, and H. Garcia-Molina. FlexRecs: expressing and combining flexible recommendations. In *Proceedings of the 35th SIGMOD international conference on Management of data*, pages 745–758, Providence, Rhode Island, USA, 2009. ACM.

[10] S. Lawrence, C. L. Giles, and K. Bollacker. Digital libraries and autonomous citation indexing. *Computer*, 32(6):67–71, 1999.

[11] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. *Computer Networks*, 33(1-6):387–401, June 2000.

[12] W. Lu, J. Janssen, E. Milios, N. Japkowicz, and Y. Zhang. Node similarity in the citation graph. *Knowledge and Information Systems*, 11(1):105–129, Jan. 2007.

[13] P. Massa and P. Avesani. Trust-Aware collaborative filtering for recommender systems. In *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, pages 275–301. 2004.

[14] S. M. Mcnee, I. Albert, D. Cosley, P. Gopalkrishnan, S. K. Lam, A. M. Rashid, J. A. Konstan, and J. Riedl. On the recommending of citations for research papers. In *In CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 116—125, 2002.

[15] S. M. McNee, J. Riedl, and J. A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI '06 extended abstracts on Human factors in computing systems*, pages 1097–1101, Montréal, Québec, Canada, 2006. ACM.

[16] S. M. McNee, J. Riedl, and J. A. Konstan. Making recommendations better: an analytic model for human-recommender interaction. In *CHI '06 extended abstracts on Human factors in computing systems*, pages 1103–1108, Montréal, Québec, Canada, 2006. ACM.

[17] D. Montgomery. *Design and analysis of experiments*. J. Wiley and Sons, New York N.Y., 5th ed. edition, 2001.

[18] P. Ogilvie and J. Callan. Experiments using the lemur toolkit. In *NIST Special Publication 500-250: Proceedings of the Tenth Text Retrieval Conference (TREC-10)*, page 103–108. NIST, 2002.

[19] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 422, Stanford InfoLab, Nov. 1999.

[20] 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*, pages 175–186, Chapel Hill, North Carolina, United States, 1994. ACM.

[21] S. Robertson and S. Walker. Okapi/Keenbow at TREC-8. In *NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC-8)*, volume 8, pages 151–161. NIST, 1999.

[22] R. Torres, S. M. McNee, M. Abel, J. A. Konstan, and J. Riedl. Enhancing digital libraries with TechLens+. In *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 228–236, Tuscon, AZ, USA, 2004. ACM.

[23] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 266–275, Washington, D.C., 2003. ACM.

[24] C. Zhai. Notes on the lemur TFIDF model. http://lemurproject.org/lemur/tfidf.pdf, Oct. 2001.